Universidad Carlos III de Madrid
Escuela Politécnica Superior
Grado en Ingeniería Informática
Mención en Computación
Planning and Learning Group

Interacción Humano-Robot con el Robot REEM sobre el Framework RoboComp

Trabajo de Fin de Grado

por

Carlos Manzano Carrasco

23 de febrero de 2016

Tutor:
José Carlos González Dorado
Cotutor:
Fernando Fernández Rebollo

TRABAJO DE FIN DE GRADO

INTERACCIÓN HUMANO-ROBOT CON EL ROBOT REEM SOBRE EL FRAMEWORK ROBOCOMP

Autor: CARLOS MANZANO CARRASCO
Tutor: JOSÉ CARLOS GONZÁLEZ DORADO
Cotutor: FERNANDO FERNÁNDEZ REBOLLO

TRIBUNAL

Presidenta: BEGOÑA SAVOINI CARDIEL
Secretaria: ISABEL ROSARIO CENAMOR GUIJARRO
Vocal: DAVID PALOMAR DELGADO

Tras el acto de defensa y lectura el día 7 de Marzo de 2016 en la **Escuela Politécnica Superior** de la **Universidad Carlos III de Madrid** (Leganés), el tribunal le otorga la siguiente **CALIFICACIÓN**:

Agradecimientos

Parecía que nunca iba a llegar este momento, pero finalmente ha llegado. Con todo el documento escrito, y como si de la gala de los Oscar se tratase, toca agradecer a aquellos que han aportado su granito de arena en el proyecto.

En primer lugar, agradecer a la empresa PAL Robotics en general, y a Franceso, Sam, Victor y Luca en particular toda la ayuda recibida, sin la cual no habría sido posible realizar este proyecto. Sus rápidas respuestas nos han sacado de más de un apuro durante la realización del trabajo.

Por supuesto, agradecer a todo el grupo del PLG de la Universidad Carlos III de Madrid el buen rollo, la ayuda y apoyo recibidos, especialmente a Fernando y a José Carlos, que viernes tras viernes nos ha aguantado hasta que se apagaban las luces para que esto llegase a funcionar. Gracias por tu dedicación, habría sido imposible terminar esto sin tu ayuda.

A mi familia, agradecerles todo el apoyo mostrado no solo ahora, sino durante toda la carrera y toda la vida. Gracias a ellos soy la persona que soy a día de hoy.

No puede faltar hueco para mis amigos y compañeros de clase, Rubén, Maroto, Gutmor, Sergio, Victor y otros tantos que me dejo, compañeros de penurias y entregas a última hora que han hecho de la carrera algo mucho más divertido y menos serio de lo que cabría esperar.

Mención especial merecen Andrea y Georgi. Georgi, amigo incondicional con el que sé que puedo contar para lo que haga falta. Tu amistad y tu apoyo son una de las cosas más valiosas que he podido encontrar en esta vida, gracias por estar ahí en las buenas y en las malas.

Por último, pero no menos importante, obligada mención a Andrea. Lo que empezó siendo una compañera de clase y ha terminado siendo una de las mejores amistades que se puedan encontrar y con la que se que podré contar para todo. Hemos llegado hasta aquí juntos y no podíamos terminar de otra forma que compartiendo esta última batalla ¡Gracias por ser como eres!

Email

carlosmc922@gmail.com

Por favor, cita este trabajo como:

Carlos Manzano Carrasco: *Interacción Humano-Robot con el robot REEM sobre el Framework RoboComp*, Trabajo de Fin de Grado, Universidad Carlos III de Madrid, 2016.

Abstract

This document presents the bachelor thesis *Human-Robot Interaction with the REEM robot using the RoboComp Framework* for the University Carlos III de Madrid, in the Polytechnic School of Leganés.

The developed project creates a new component for the NAOTherapist architecture that replaces the original NAO robot by the REEM robot. The new component maintains the compatibility with the rest of the architecture and adapts all existing functionality. This involves adapting the robot movement, developing a graphical interface to solve the morphological and functional differences between the two robots and adding elements that encourage user interaction, such as playing audio and text or predefined animations.

In this document the whole process of the development of this component is described, as well as the performed evaluation based on the Simon memory game, which seeks to promote human-robot interaction. An overview is available in Appendix B.

Keywords

Human-Robot Interaction, Robotic Framework, Retargeting, 3D Sensor, REEM robot, ROS, RoboComp.

Resumen

El presente documento muestra el Trabajo de Fin de Grado realizado *Interacción Humano-Robot con el robot REEM sobre el Framework RoboComp* en la Universidad Carlos III de Madrid, en la Escuela Politécnica Superior de Leganés.

El proyecto desarrollado crea un nuevo componente dentro de la arquitectura NAOTherapist que sustituye el robot NAO original por el robot REEM. Este componente mantiene la compatibilidad con el resto de la arquitectura, además de adaptar toda la funcionalidad ya existente. Esto implica adaptar el movimiento a la configuración de articulaciones del nuevo robot, desarrollar una interfaz gráfica que solucione las diferencias entre ambos robots a nivel morfológico y funcional, y agregar elementos que favorezcan la interacción con el usuario, tales como reproducción de audio y texto o animaciones predefinidas.

En este documento se describe todo el proceso de desarrollo de este componente, además de una evaluación del mismo basada en el juego de memoria Simon, con el que se pretende fomentar la interacción humano-robot del sistema.

Palabras clave

Interacción Humano-Robot, Framework Robótico, Retargeting, Sensor 3D, robot REEM, ROS, RoboComp.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se presenta el contexto y estructura que sigue este proyecto, en el que se desarrolla un nuevo componente para el proyecto NAOTherapist [González et al. 2015], utilizando el robot REEM[1]. Este componente sustituirá el robot NAO[2] por el robot REEM, manteniendo la compatibilidad con toda la arquitectura y la máxima funcionalidad posible respecto al NAO. En este primer apartado se sitúa el proyecto así como los objetivos concretos del mismo, con el fin de clarificar el trabajo que se va a realizar y lo que se quiere conseguir con su realización.

Pese a que el proyecto NAOTherapist está enfocado a las terapias de rehabilitación, este trabajo se enfoca desde un punto de vista diferente, en el que se utilizará la arquitectura NAOTherapist para fomentar la interacción humano-robot. Potenciar este componente interactivo del robot con el ser humano permite que la arquitectura NAOTherapist pueda aplicarse en otros campos, además del ámbito hospitalario. Para este proyecto se utilizará una adaptación del conocido juego de memoria Simon, que originalmente consiste en recordar una secuencia de colores y repetirla. Por cada acierto, esa secuencia se irá haciendo más larga hasta que el jugador falle.

El desarrollo de este proyecto requiere de la integración e instalación de dos plata-

[1] http://pal-robotics.com/es/products/reem/ Accedido por última vez el 11/01/2016

[2] http://www.aldebaran.com/en/humanoid-robot/nao-robot/ Accedido por última vez el 11/01/2016

formas robóticas diferentes en una arquitectura ya existente, además de comunicación con la empresa creadora del robot REEM para obtener herramientas clave para el desarrollo. Debido a la envergadura del desarrollo del proyecto, que requiere también pruebas en laboratorio, el trabajo se ha realizado de forma conjunta entre Andrea Haro Casas y el autor de este proyecto. Ambos alumnos enfocan su trabajo hacía distintos dominios, pero debido a una primera parte de desarrollo que inevitablemente es común y a las numerosas dependencias entre ambos desarrollos, existe al menos un grado de implicación de un 20 % en todas las partes realizadas. No obstante, se trata de un trabajo complementario, en el que la unión de cada parte del desarrollo conforma el nuevo componente, denominado ReemComp. A lo largo del presente documento se detallará la realización del trabajo realizada por el autor de este documento, estando reflejada la repartición de cada persona del equipo de desarrollo en la Tabla 1.1.

1.1. Contexto

El proyecto NAOTherapist es un sistema de diseño y supervisión de terapias de rehabilitación de extremidades superiores. En primer lugar se planifican los ejercicios que se van a hacer, y un robot autónomo y humanoide guía al paciente, indicándole las posturas que debe realizar y corrigiendo los movimientos en caso de que éstos no sean correctos[3]. El proyecto original se realizó con un robot NAO como el que se puede ver en la Figura 1.1.

Esta arquitectura se explicará en detalle en el Capítulo 2 del presente documento, en el que se expondrán todas las partes de la misma así como el componente concreto que se ha desarollado para este proyecto.

El campo de la Robótica Social Interactiva (SIR por sus siglas en inglés, *Social Interactive Robotics*) estudia la posibilidad de crear robots autónomos capaces de crear cualquier tipo de interacción y comunicación tanto con humanos como con el entorno

[3] https://youtu.be/75xb39Q8QEg Accedido por última vez el 11/01/2016

Figura 1.1: Robot NAO.

que los rodea. Un primer ejemplo de esto podemos verlo en las famosas tortugas creadas por Walter Grey en los años 40 [Fong et al. 2003], en el que se diseñó uno de los primeros robots considerados inteligentes, capaces de interactuar y esquivar objetos. Desde ese momento se han desarrollado todo tipo de robots con el objetivo de mejorar dicha interacción, como por ejemplo los robots mayordomo [Graf et al. 2009] o robots capaces de tocar música junto a músicos humanos siguiendo el ritmo [Weinberg et al.]. Las ventajas de este uso de robots interactivos dependen mucho del ámbito en el que se apliquen, pero se ha demostrado que la interacción mediante el uso de robots ha tenido resultados satisfactorios en campos como terapias para el tratamiento del autismo [Cabibihan et al. 2013].

Por otro lado, Simon es un juego electrónico de memoria en el que se utiliza un disco con cuatro colores diferentes como el que se puede ver en la Figura 1.2. El juego creado por Ralph Baer y Howard J. Morrison en 1978 tiene cuatro cuadrantes, con los colores rojo, azul, verde y amarillo, cada uno de los cuales reproduce un sonido diferente al ser pulsado.

Figura 1.2: Juego Simon original.

El objetivo del juego es conseguir recordar una secuencia lo más larga posible, en la que se irán encendiendo aleatoriamente los cuadrantes a la vez que se reproduce un sonido característico de cada cuadrante. Se empieza con un único color encendido, y el usuario debe repetir la secuencia. A medida que el usuario va avanzando, la secuencia de colores a recordar cada vez es más larga, además de mostrarse a una velocidad mayor aumentando así el nivel de dificultad del juego.

El nombre del juego proviene del tradicional juego *Simon dice*, en el que varias personas se reúnen en torno a un participante que hace las veces de Simon (encargado de dirigir el juego) y el resto de participantes realizan lo que Simon dice. La mecánica del juego es muy simple, si Simon dice **"Simon dice baila"**, el resto de participantes deben bailar o quedar eliminados. En cambio, si únicamente dice **"Salta"**, los participantes no deben saltar o perderán. El juego se termina cuando solo queda un participante siguiendo las órdenes de Simon.

El juego ha tenido muchas variantes a partir del original, con mayor número de colores y sonidos (ver Figura 1.3), incluso el tiempo de pulsación de cada color. No es lo mismo una pulsación rápida del rojo que una larga del mismo color por ejemplo.

Figura 1.3: Versión modificada de Simon con más colores añadidos.

Por último, destacar que la arquitectura NAOTherapist es lo suficientemente general como para poder usarse con otros dominios [Turp et al. 2015] y con otros robots, como es el caso del presente documento. Este trabajo consiste en conseguir utilizar la arquitectura NAOTherapist con el robot REEM orientando el uso del robot REEM al entretenimiento. Se quiere dotar al robot de una parte audiovisual que potencie la parte interactiva del robot, como pueden ser juegos o cualquier tipo de actividad de animación. El dominio sobre el que se centra el proyecto es el juego de memoria Simon. En este proyecto se utilizará una versión del juego en la que el objetivo no es recordar una secuencia de colores, sino una secuencia de movimientos que irá indicando el robot y que el usuario deberá de seguir.

A diferencia de NAOTherapist, que está desarrollada utilizando el *framework* robótico *RoboComp*, REEM se controla mediante otro *framework*, *ROS*. Ambas plataformas sirven para el control de robots y se detallarán en el Capítulo 2 de este documento. Una de las principales dificultades de este proyecto consiste en establecer un mecanismo en el que ambos *framework* puedan convivir sin ningún tipo de incompatibilidad.

1.2. Motivación

El objetivo de este proyecto es crear un nuevo componente para la arquitectura NAOTherapist, para la utilización del robot REEM en dicha arquitectura y su aplicación en el dominio alternativo de Simon. Se desea que este componente tenga la máxima funcionalidad posible teniendo en cuenta la funcionalidad que tiene el componente original basado en el robot NAO. Para ello se pretende desarrollar tanto la parte orientada al movimiento del robot como toda la parte audiovisual del mismo. Conseguir que el robot REEM sea 100 % compatible con la arquitectura NAOTherapist es el fin último de este desarrollo.

Con el fin de maximizar esta compatibilidad se ha hecho necesario incluir una interfaz gráfica que emule los botones físicos del robot NAO y los leds de los ojos, fundamentales para que el usuario tenga un *feedback* sobre las posturas realizadas. En la misma linea es necesario incluir la reproducción de audio tanto con ficheros existentes como a través de texto, ya que potencia en gran medida la interacción con el usuario. También es importante la realización de un sistema de *retargeting* con el objetivo de conseguir un movimiento lo más acorde a la realidad posible.

La arquitectura NAOTherapist está enfocada a la rehabilitación en hospitales, por lo que una de las consecuencias de la realización de este proyecto es su aplicación en el ámbito hospitalario. De acuerdo a la motivación original de NAOTherapist, las enfermedades principalmente tratadas afectan a las extremidades superiores, por lo que el uso de REEM (ver Figura 1.4) puede ser beneficioso debido a que es del tamaño de una persona adulta. En ningún caso se pretende que sea el sustituto del NAO, el objetivo es que existan la mayor cantidad de alternativas posibles. Dependiendo de cada paciente puede ser interesante utilizar uno u otro robot, lo que se traduce en que la arquitectura pueda ser aplicable a un mayor número de entornos y problemas.

Las ventajas de utilizar una terapia de este tipo con robots son amplias. En primer lugar, se reduce en gran parte el coste asociado a personal especializado durante las

Figura 1.4: Robot REEM.

terapias. Aunque no es un sistema completamente autónomo, ya que su objetivo no es sustituir al terapeuta, sino facilitar su trabajo y complementarlo. Por otra parte, como se ha mencionado en el punto anterior, las enfermedades para las que ha sido desarrollado no tienen cura conocida a día de hoy. Eso implica que las terapias deben realizarse de por vida, y generalmente desde edades muy tempranas. El uso de robots humanoides que guíen las terapias puede ser una buena forma de que los niños no pierdan el interés en las mismas [Matarić et al. 2007]. Por ello el componente interactivo robot-humano es uno de los objetivos a desarrollar. El uso de Simon en las terapias puede también tener ventajas para su uso terapéutico, además de su uso en el entretenimiento [Turp et al. 2015]:

- Plantear la terapia como un juego puede ser muy útil a la hora de que los niños no pierdan el interés, por lo que ayudaría al terapeuta a que los niños continúen realizando las terapias mejorando ostensiblemente su calidad de vida.

- Simon es un juego de memoria, gracias a lo cual el sistema puede ser aplicable a otro tipo de enfermedades como el Alzheimer. Además de ejercitar la musculatura, se podrán desarrollar terapias que ayuden a la estimulación de la memoria.

1.3. Objetivos del trabajo

Los objetivos concretos de este trabajo se pueden resumir del siguiente modo:

1. Integración del *framework* ROS (*Robot Operating System*) junto a la arquitectura NAOTherapist (que usa el *framework* RoboComp), imprescindible para que el robot REEM se pueda comunicar con el resto de componentes de la arquitectura NAOTherapist.
2. Realizar la ejecución de movimientos del robot REEM en el simulador Gazebo a través de comandos simples en Python.
3. Partiendo de un esqueleto humanoide extraído de la cámara de una Kinect, implementar el retargeting de los ángulos de cada articulación para su correcto movimiento.
   a) Seleccionar las articulaciones sobre las que se podrá realizar un retargeting (adaptación del valor de los ángulos del esqueleto obtenido del usuario a los ángulos propios del robot, ver en detalle en la sección 2.4) en función de las particularidades del robot REEM.
   b) Transformar los ángulos de manera que se realice el mismo movimiento tanto en el esqueleto de la Kinect como en el robot REEM.
4. Implementar una interfaz gráfica que se pueda representar en la pantalla integrada del REEM. Se simularán una serie de pilotos LED que servirán para dar información visual a la persona que esté interactuando con el robot, además de un conjunto de botones que simulan los botones físicos del robot NAO.
5. Incluir reproducción de audio y un sistema *Text-to-Speech* para facilitar la comunicación oral del robot con el usuario.

6. Integración del sistema descrito en los puntos anteriores aplicado al dominio adaptado del juego clásico *Simon*.

1.4. Estructura de la memoria

La memoria se ha estructurado de forma que parte del contenido más general y a medida que se avanza en la misma se centra en lo mas específico del trabajo. El primer capítulo, al que corresponde este apartado, se centra en la introducción y motivación general del proyecto, así como los objetivos que se desean cumplir con su realización. El Capítulo 2 presenta el estado de la cuestión, en la que se incluyen los principales elementos teóricos y técnicos sobre las soluciones que se han adoptado en este trabajo. El Capítulo 3 se centra en el análisis y el diseño del sistema. En este apartado se recogerá toda la especificación de requisitos del sistema, así como los casos de uso para comprender el funcionamiento del sistema. En el Capítulo 4 se tratará la implementación propiamente dicha del mismo, en el que se mostrarán las partes principales de este trabajo: integración del robot REEM en la arquitectura RoboComp, implementación del sistema de retargeting para adaptar el movimiento del robot al resto de componentes, así como el desarrollo de la interfaz gráfica y funcionalidad de reproducción de audio y text-to-Speech.

Para probar que los objetivos se han cumplido, en el Capítulo 5 se presentará toda la evaluación del sistema, que contendrá una serie de experimentos y pruebas que demostrarán si los objetivos se han cumplido. El Capítulo 6 contiene una planificación de todo el desarrollo del proyecto, además del presupuesto destinado a este trabajo. Por último, en el Capítulo 7 se discuten las conclusiones y trabajos futuros de todo lo explicado anteriormente.

En este apartado se quiere recoger también la carga de trabajo asociada al equipo de desarrollo (tanto al trabajo realizado por Andrea Haro como al del autor de este trabajo de fin de grado). Como ya se ha mencionado en la introducción, la carga de

TAREA	Carlos Manzano	Andrea Haro
Análisis del sistema	50%	50%
Diseño del sistema	50%	50%
Implementación del sistema		
Integración de ROS en NAOTherapist	50%	50%
Comandos simples en Python	50%	50%
Implementación retargeting	40%	60%
Desarrollo interfaz	70%	30%
Comunicación de ReemComp con la interfaz	80%	20%
Reproducción de audio	100%	0%
Text-to-Speech	100%	0%
Evaluación del sistema		
Dominio terapias	0%	100%
Dominio Simon	100%	0%

Tabla 1.1: Reparto de trabajo entre los miembros del equipo de desarollo.

trabajo y las numerosas dependencias de ambos proyectos han hecho que el proyecto se haya realizado de forma conjunta. Pese a estar enfocados a dominios distintos, se ha necesitado colaboración en el desarrollo e integración de diversos componentes por parte de ambos alumnos. Esta colaboración y reparto de trabajo queda recogida en la Tabla 1.1, en la que se muestra mediante porcentajes orientativos la dedicación de cada alumno al desarrollo del proyecto. Destacar que debido a la divergencia entre proyectos a fecha de entrega de este proyecto el trabajo de Andrea Haro aún no ha sido presentado.

Los apartados marcados con un 50 % indican que la carga de trabajo en ese punto ha sido equitativa, no que el trabajo realizado haya sido el mismo. En el resto de casos en los que el porcentaje no es igual, la implicación de cada alumno ha sido diferente dependiendo del enfoque de cada uno de los dos proyectos. Salvo en la experimentación se ha necesitado colaboración de ambas partes en todos los puntos, principalmente debido a la necesidad de integrar todas las mejoras de cada alumno al sistema, para que ambos pudieran contar con toda la funcionalidad desarrollada.

Capítulo 2

Estado de la cuestión

En este capítulo se muestra un análisis del estado actual en el que se encuentra el marco en el que se enfoca este proyecto. Además, se detallan las principales herramientas y tecnologías utilizadas durante su desarrollo con el fin de facilitar la comprensión de la implementación explicada en el Capítulo 4.

2.1. HRI: Interacción humano-robot

En esta sección se realiza una breve introducción general a la robótica orientada al entretenimiento para después profundizar en los aspectos de la Interacción Humano-Robot, ya que se trata de uno de los principales elementos a potenciar en este proyecto.

El término *robot* surge por primera vez en 1921 en la obra *Rossum's Universal Robots* de Karel Capek [Capek 1921]. Etimológicamente, el término proviene del checo *robota*, que significa trabajo forzada o servidumbre. Desde mediados del siglo XX el campo de la robótica ha estado en constante crecimiento, desarrollando y perfeccionando todo tipo de robots para multitud de entornos y situaciones, desde brazos robóticos utilizados en cadenas de montaje hasta robots capaces de explorar otros planetas, como los *rover* de la NASA. Una parte de la robótica que va estrechamente ligada al desarro-

Figura 2.1: Robot AIBO.

llo de este proyecto es la rama dedicada al entretenimiento. En esta rama se desarrollan robots que fomenten la interacción humano-robot en todo tipo de ambientes, de la que se hablará en el siguiente subapartado. Además, por norma general constan de un fuerte componente visual capaz de mantener la atención y el interés del usuario con el que interactúan, así como ser capaces de llegar a provocar emociones humanas. Se han llegado a realizar estudios sobre como afecta a las emociones y calidad de vida la interacción de una persona con un robot mascota como el AIBO de Sony (ver Figura 2.1) [Fujita 2004].

La demanda de este tipo de robots de entretenimiento es muy alta, pero aún existen problemas a su alrededor. Un ejemplo claro se encuentra en los robots del hogar. Estos robots deben interactuar físicamente con el usuario y su entorno, por lo que cualquier error de reconocimiento o control puede implicar que el usuario resulte herido o se produzcan daños en el entorno. El estado ideal implicaría crear un robot "perfecto", sin ningún error de control ni reconocimiento, pero la situación actual

está muy lejos de este estado. Utilizar robots de menor tamaño y peso mitigaría los problemas mencionados, pero conlleva nuevos problemas a la hora de realizar tareas físicas, por ejemplo.

Los sistemas de reconocimiento del entorno como la visión y la toma de decisiones también tienen mucho camino por recorrer. Actualmente estas líneas de investigación son complicadas y a menudo se intentan acometer con técnicas de inteligencia artificial, como la Planificación Automática [Ghallab et al. 2004]. En definitiva, es un campo todavía en desarrollo con muchas expectativas por delante.

Una de las maneras de estimular el desarrollo de estas líneas de investigación es mediante la realización de competiciones, entre las cuales cabe destacar el proyecto RoboCup. Su misión es crear un equipo de robots humanoides autónomos que en el año 2050 sea capaz de ganar al equipo campeón mundial de fútbol humano. Inicialmente esta iniciativa utilizó robots AIBO, para después ser sustituidos por un equipo de robots NAO [Veloso et al.].

En el proyecto descrito en este documento se aplica la arquitectura NAOTherapist para hacer una versión adaptada del juego Simon, cuyo principal objetivo es el entretenimiento.

La interacción humano-robot (HRI, por sus siglas en inglés) se encarga de estudiar las relaciones entre humanos y robots. Esta interacción en el fondo está basada en la comunicación humana, por lo que muchos aspectos del HRI son continuaciones del estudio de las comunicaciones humanas. Se trata de un campo relativamente joven e interdisciplinar, presente en robótica, ingeniería, psicología o investigación de comportamientos sociales, entre otros [Dautenhahn 2007].

El origen ético del problema de la interacción entre humanos y robots existe desde la creación de los primeros robots. En la obra del científico y escritor Isaac Asimov ya se plantearon las famosas tres leyes de la robótica. Pese a ser relatos de ficción, de estas tres leyes se desprende el principal problema de la cuestión. Es necesario establecer unos principios claros para fomentar la interacción, siempre y cuando se

Figura 2.2: Dos robots NAO durante un partido de la RoboCup.

mantenga la integridad del ser humano. Con el avance de las tecnologías los robots cada vez muestran comportamientos más proactivos, por lo que es fundamental sentar unas bases para garantizar la seguridad del ser humano. En 2011, el Consejo de Investigación de Ingeniería y Ciencias Físicas (EPSRC por sus siglas en inglés) junto con el Consejo de Investigación de Artes y Humanidades de Gran Bretaña(AHRC por sus siglas en inglés) publicaron un conjunto de principios éticos tanto para diseñadores como para usuarios de robots en el mundo real[1]. Cabe destacar que el proyecto desarrollado en este documento no implica contacto físico con el usuario, por lo que no hay riesgos para el ser humano en este aspecto.

Entre las líneas de estudio que se encuentran en el campo de la interacción humano-robot, principalmente se están desarrollando métodos para la percepción de humanos, con modelos de visión 3D del entorno y de su situación en el mismo. Para ello se usan

[1] http://www.webcitation.org/6RJYLsU8m Accedido por última vez el 29/01/2016

múltiples sensores como los proporcionados por el controlador de juego creado por Microsoft, Kinect (Ver detalle en Sección 2.3.3). Estos sensores garantizan reconocimiento visual, de gestos y de voz que pueden ser usados para controlar el robot. Una técnica de aprendizaje que aplica estos sistemas es el aprendizaje por demostración en agentes exploradores, recopilando información del enterno y aprendiendo una política en función de los ejemplos obtenidos [Argall et al. 2009].

De la misma manera, se siguen líneas de estudio que mejoren la capacidad de movimiento de los robots, mejorándolos para que el movimiento sea más suave, pesen menos y sus movimientos sean más precisos y estables. También se realiza el estudio de modelos cognitivos que mejoren el comportamiento de los mismos, como es el caso PELEA (basado en Planificación Automática) [Alcázar et al.].

Centrado en el aspecto interactivo del robot, se pretende mostrar al robot como una entidad autónoma capaz de plantear sus propias metas en base a motivaciones o emociones, donde la interacción con el ser humano le permite completar dichas necesidades (necesidades sociales).

Para el ser humano, el principal fin de la robótica social es que le sirva para completar sus tareas y ayudar en el día a día. En este sentido se ha comprobado que el diseño del robot influye notablemente en la percepción que se tiene del mismo, por lo que en la actualidad se realizan todo tipo de diseños con el objetivo de que el ser humano los encuentre como algo “amigable” o “agradable”.

En el Capítulo 1 de este documento se presentó el término *Robot interactivo social (SIR)*, en el que la interacción social juega un papel clave para el HRI [Fong et al. 2003]. Este conjunto de robots tiene como características principales expresar y percibir emociones, comunicación con diálogos de alto nivel, establecer o mantener relaciones sociales y utilizar un lenguaje corporal natural. En resumen, los SIR pretenden mantener un comportamiento similar al que tendría el propio ser humano.

Es un área todavía en desarrollo, ya que a día de hoy no existe el robot que plantee una interacción perfecta, pero los avances son notorios. En el desarrollo de este proyecto se pretende aportar valor a dicho campo potenciando la interacción del robot REEM con el ser humano.

2.2. Proyecto NAOTherapist

El objetivo del proyecto THERAPIST consiste en crear un sistema de terapias de rehabilitación de las extremidades superiores utilizando un robot humanoide autónomo que se encargue de guiar las terapias. Inicialmente el proyecto THERAPIST se realiza con el robot Ursus [Suárez Mejías et al. 2013], pero debido a su falta de autonomía al ser un robot totalmente teleoperado se expande el proyecto para dotarle de inteligencia artificial (utilizando Planificación Automática). Así surge el proyecto THERAPIST tal y como se conoce actualmente.

El proyecto THERAPIST[2] es un proyecto del Plan Nacional de Investigación (TIN2012-38079-C03-01) en el que trabajan varios grupos de investigación de diversas universidades españolas (entre ellas la Universidad Carlos III de Madrid) en colaboración con un grupo experto de médicos y terapeutas del Hospital Universitario Virgen del Rocío de Sevilla. El objetivo de este proyecto es desarrollar terapias de rehabilitación de extremidades superiores mediante el uso de robótica social de asistencia, en la que se utilizará un robot para realizar parte de la labor del terapeuta. Gracias a la colaboración del Hospital Universitario Virgen del Rocío se tiene acceso tanto al protocolo terapéutico interno del hospital como a pacientes reales. Los pacientes que participan en este proyecto son menores de edad entre 3 y 14 años que padecen Parálisis Cerebral Infantil o Parálisis Braquial Obstétrica con alteraciones motrices en las extremidades superiores.

La Parálisis Braquial Obstétrica (PBO) es una debilidad o pérdida de movimiento

[2] http://www.therapist.uma.es/ Accedido por última vez el 11/01/2016

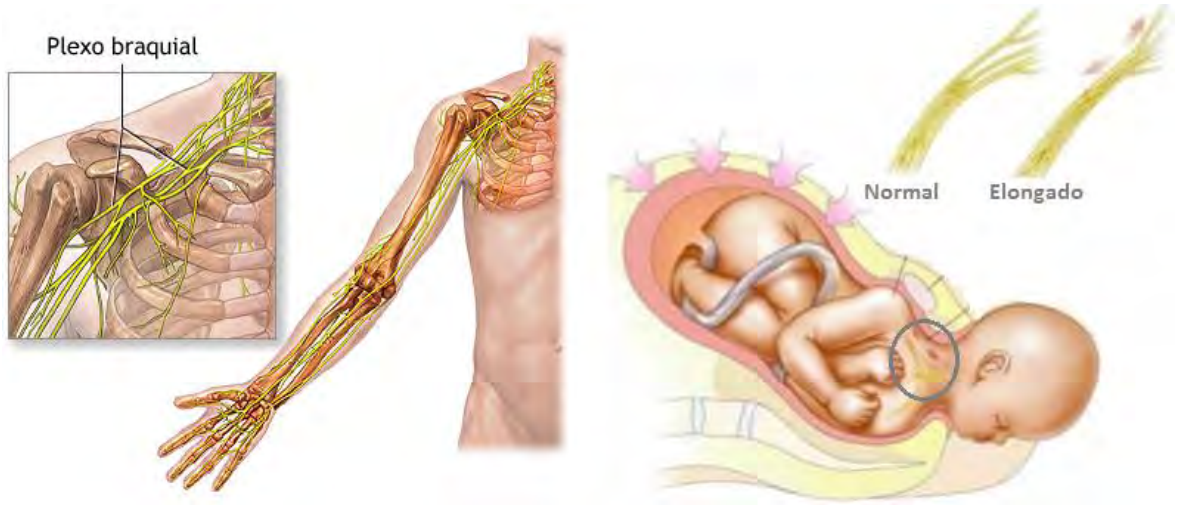

Figura 2.3: Plexo braquial y su elongación durante el parto.

de las extremidades superiores producida cuando el conjunto de nervios alrededor del hombro (denominado plexo braquial) se daña durante el nacimiento (ver Figura 2.3). Gracias a los avances de la medicina y la mejora de las técnicas durante el parto, esta enfermedad se ha reducido drásticamente. Aun así, se estima que 1,5 de cada 1000 nacimientos presentan este problema. Este problema no es exclusivo del parto, ya que otras causas como los accidentes de tráfico pueden producir esta lesión. En cualquier caso, es necesaria mucha rehabilitación física para recuperar movilidad en el brazo afectado [Limthongthang et al. 2013].

La Parálisis Cerebral Infantil (PCI) es una enfermedad cerebral no progresiva que impide tener un completo control de las funciones motoras [Krägeloh-Mann et al. 2009]. Como ocurre con la Parálisis Braquial Obstétrica, esta afección se produce principalmente por complicaciones durante el parto, aunque también se puede producir a causa de un accidente. Un 60 % de los niños con deficiencias motoras están diagnosticados con esta enfermedad. Aunque no es una afección común, existen suficientes casos como para no llegar a ser considerada una enfermedad rara, teniendo 2 o 3 casos de cada 1000 nacimientos cuyos síntomas coinciden con esta enfermedad [Castelli 2011].

En ambos enfermedades no existe cura conocida , por lo que un paciente afectado

deberá convivir con ellas toda la vida. La rehabilitación juega un papel muy importante en estos casos, ya que si bien no se puede llegar a curar completamente, sí se puede mejorar enormemente la calidad de vida de la persona. Gracias a una buena rehabilitación el paciente es capaz de fortalecer la musculatura y acostumbrar al cerebro al modo de mover las extremidades afectadas.

NAOTherapist surge como una escisión del proyecto THERAPIST en la que se incluye la compatibilidad con el robot NAO. Como ya se introdujo en el Capítulo 1 del presente documento, NAOTherapist [González et al. 2015] es una arquitectura robótica que contiene un sistema de diseño y supervisión de terapias de rehabilitación. Se planifica una serie de ejercicios para realizar el entrenamiento, y posteriormente un robot humanoide (NAO en el proyecto original, adaptado a REEM en este proyecto) guiará al paciente enseñándole los movimientos que tiene que realizar y corrigiendo su postura en caso de no realizarla correctamente. En la Figura 2.4 se puede ver la arquitectura al completo, de la que se detallará a continuación el funcionamiento y utilidad de cada componente. Destacar que el nuevo componente a desarrollar sustituirá al componente *NAO Robot* manteniendo la compatibilidad con toda la arquitectura, debido a su diseño independiente de la plataforma robótica.

Este proyecto podría englobarse en los llamados *robots sociales de asistencia* (SAR, por sus siglas en inglés). Esta categoría incluye los robots que ofrecen algún tipo de asistencia mediante interacción social, sin llegar al contacto físico. Se ha demostrado que las terapias de rehabilitación en las que un terapeuta realiza una serie de ejercicios con el paciente son uno de los métodos más efectivos de rehabilitación. Las aproximaciones de SAR entre paciente y robot autónomo para asistencia terapéutica están actualmente entre las principales líneas de investigación [Eriksson et al.].

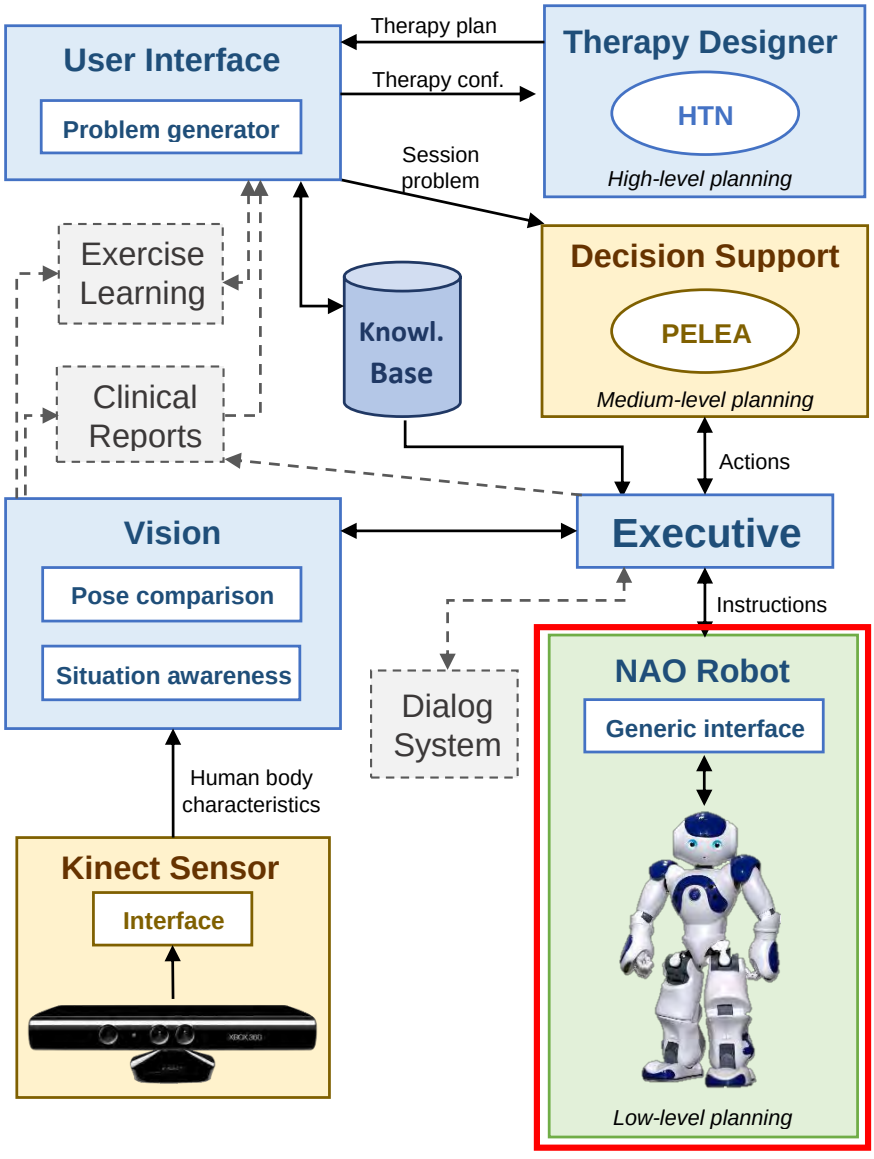

Figura 2.4: Arquitectura NAOTherapist.

Los principales componentes de la arquitectura NAOTherapist son:

- **Diseñador de terapias (*User Interface, Therapy Designer*):** provee una interfaz gráfica para configurar la terapia a seguir en función del diagnóstico del paciente. Antes de la rehabilitación, se configura la terapia y se envía al diseñador de terapias como un problema de Planificación Automática [Ghallab et al. 2004]. La Planificación Automática es una disciplina que se encarga de la generación de planes, típicamente ejecutados por un robot o agente. Esta planificación se genera en base a un estado actual del mundo, una meta a cumplir y un conjunto de acciones disponibles. El diseñador de terapias considera todos los ejercicios disponibles en la base de datos e incluye una parte de los mismos distribuidos en tres fases (calentamiento, entrenamiento, enfriamiento), siendo más dura la parte central de la terapia. Si no hay ejercicios disponibles, el modelo podrá sugerir uno nuevo que cumpla con los objetivos terapéuticos. Este modulo está considerado como un *Clinical Decision Support System (CDDS)*, cumpliendo los criterios médicos para la planificación de terapias.
- **PELEA (*Decision Support*):** PELEA (*Planning, Execution and Learning Architecture*) [Alcázar et al.] es una subarquitectura pensada para la integración de planificación, ejecución, monitorización, replanificación y aprendizaje en robótica, aunque no es exclusiva de este campo. Tiene un diseño modular, pero solo se utilizan 3 módulos en este componente de NAOTherapist (ver Figura 2.5). PELEA se encarga de planificar un plan de acciones y devolver una a una las acciones del mismo [Hoffmann 2003]. Estas acciones tienen unos efectos que deben cumplirse, en caso de no hacerlo se replanifica y se genera un nuevo plan de acciones. Atendiendo al diagrama de funcionamiento, Execution recibe el estado del mundo actual y lo envía a Monitoring. Si el estado recibido coincide con el almacenado en *infoMonitoring*, se devuelve a Execution la acción planificada. En caso de que el estado no coincida, Monitoring pide a Decision Support un nuevo plan, que vuelve a generar un nuevo plan de acciones válido y lo devuelve a Moni-

Figura 2.5: Diagrama de funcionamiento de PELEA.

toring, que a su vez devuelve la acción a Execution. Este módulo está conectado al modulo Ejecutivo de la arquitectura NAOTherapist, que será el encargado de pedir y realizar las acciones.

- **Ejecutivo (*Executive*):** el componente ejecutivo se encarga de manejar las acciones a realizar. Ejecutivo pide la acción a PELEA, que se encarga de planificar y devolver una acción. Una vez que se obtiene esta acción se envían las instrucciones necesarias para ejecutar la acción al robot. Para poder pedir una acción Ejecutivo debe tener una visión clara del estado actual del mundo, por lo que haciendo uso del componente de Visión se recoge información del paciente sobre una serie de predicados tales como *identified-patient* o *correct-pose*. En función de los datos recogidos por los sensores se forma un estado del mundo con una serie de predicados en formato PDDL que se envían a PELEA para la planificación de la próxima acción.
- **Visión (*Vision*):** componente utilizado para obtener información sobre la pose y el estado del paciente. Gracias al uso de los sensores de la Kinect se obtienen datos característicos del paciente, como el esqueleto del cuerpo, la posición de las manos y el reconocimiento de puntos de expresión de la cara. El componente está formado principalmente por dos elementos:

  - **Pose comparison:** basándose en el esqueleto obtenido por la Kinect, se calcula la diferencia entre los valores del esqueleto de la pose que se está realizando y la pose que está poniendo el paciente en ese momento. Esto permite determinar el grado de diferencia y corregir al paciente en caso de que la

diferencia supere un umbral determinado.

- **Situation awareness:** detecta cuando el paciente está en el área de entrenamiento, si está de pie o sentado e incluso distraído. Esto permite que el robot pueda llamar la atención del paciente para proseguir con la terapia.

- **Interfaz robot (*NAO Robot*):** este componente integra toda la funcionalidad que se ha desarrollado en este trabajo de fin de grado. Es el componente del robot propiamente dicho, y se encarga de recibir las acciones que debe realizar del Ejecutivo y enviar al robot las instrucciones necesarias para ejecutar dicha acción. Esta interfaz ha sido completamente sustituida en este trabajo para que la arquitectura se pueda utilizar con el robot REEM, en lugar del NAO, sin cambiar ningún otro componente de la misma. Esto demuestra que la arquitectura es fácilmente generalizable, ya que con el cambio de un único componente se mantiene el funcionamiento total de la arquitectura.

Dialog System, Clinical Reports y *Exercise Learning* son elementos de la arquitectura que se encuentran en proceso de desarrollo y no se utilizan, por lo que no se hace referencia en este documento.

2.3. Tecnologías utilizadas

En este apartado se detallarán las principales herramientas y tecnologías utilizadas en el desarrollo e implementación del proyecto. El capítulo de implementación está fuertemente basado en estas tecnologías y algoritmos, por lo que su lectura facilita mucho la comprensión del Capítulo 4.

2.3.1. Robot REEM

El robot Reem es un robot humanoide creado por la empresa de robótica española Pal-Robotics [3]. Esta empresa formada por un equipo de ingenieros se encarga de diseñar y personalizar robots humanoides de servicio capaces de navegar autónomamente y desempeñarse en entornos reales. Para la realización de este proyecto se ha utilizado el modelo Reem, de 1,70 m. de estatura y con un peso de 100 kg como el que se puede ver en la figura 1.4.

Se trata de un robot articulado que cuenta con diferentes motores que le permiten mover brazos, torso, cabeza y manos. Pese a no tener piernas tiene una base móvil con ruedas que le permite desplazarse sin problemas, además de tener una mayor estabilidad gracias a la sólida base.

El robot cuenta con 34 grados de libertad (7 en los brazos, 7 en las manos, 2 en el torso, 2 en la cabeza y 2 en la base móvil) que le dan una amplia capacidad de movimiento[4]. Para el desarrollo de este proyecto se han utilizado las articulaciones relacionadas con los brazos para la realización del retargeting, mientras que las manos, el torso y la cabeza se han utilizado en la inclusión de las animaciones. En las Figura 2.6 se puede observar dichas articulaciones en sus valores máximo y mínimo, valores que se detallan en la Tabla 2.1.

Hay que destacar que el giro del hombro no es perpendicular respecto al torso, como se puede ver en la Figura 2.7. En azul se puede ver el ángulo sobre el que trabaja el NAO, y en rojo la desviación que produce la morfología propia del robot REEM. Esto provoca que al abrir el brazo 90º no queda perpendicular como si ocurre en el NAO, por lo que se han realizado transformaciones para corregir este desfase (explicadas en el Capítulo 4 de este documento).

[3] http://pal-robotics.com/es/company/ Accedido por última vez el 26/01/2016
[4] https://youtu.be/SRcu8nUzTdE Accedido por última vez el 26/01/2016

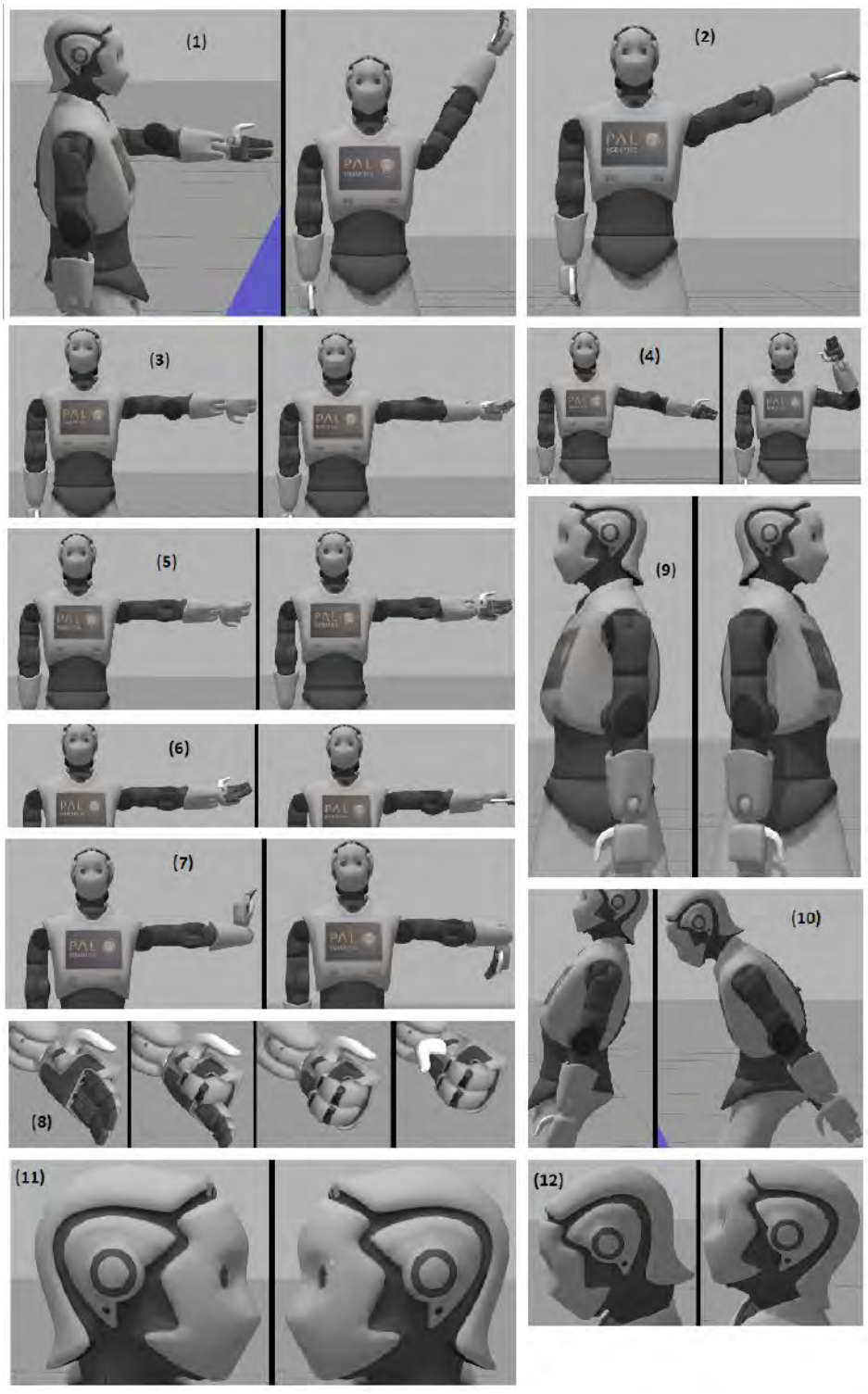

Figura 2.6: Articulaciones del robot REEM.

FIGURA	ARTICULACIÓN	MÁXIMO (º)	MÍNIMO (º)
(1)	GiroHombro	118º	-13º
(2)	AperturaHombro	178º	-44º
(3)	GiroCodo	156º	-134º
(4)	AperturaCodo	123º	0º
(5)	GiroAntebrazo	118º	-118º
(6)	GiroMuñeca	88º	-88º
(7)	AperturaMuñeca	88º	-88º
(8)	Dedos	Pulgar: 88º Resto: 248º	Pulgar: 0º Resto: 0º
(9)	GiroTorso	73º	-73º
(10)	InclinaciónTorso	34º	-13º
(11)	GiroCabeza	73º	-73º
(12)	InclinaciónCabeza	34º	-13º

Tabla 2.1: Valores mínimo y máximo de las articulaciones de REEM.

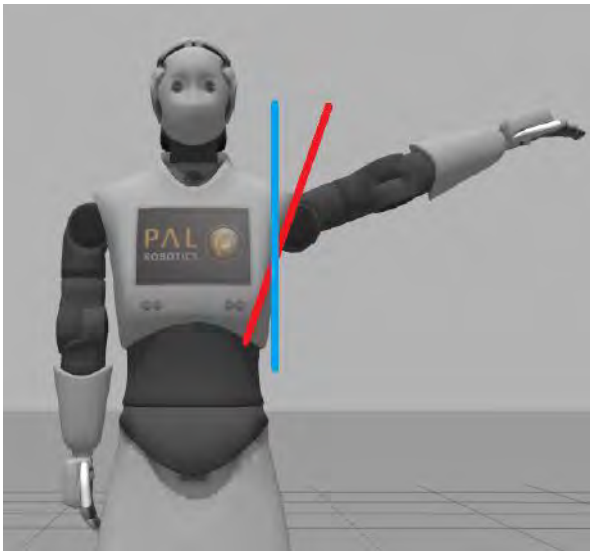

Figura 2.7: En azul, el plano sobre el que se calculan los ángulos del NAO, en rojo el cálculo de REEM.

La arquitectura del robot permite que sea capaz de sujetar objetos de tamaño bastante reducido con las manos, lo cual es un punto a favor a la hora de fomentar la interacción con personas u otros elementos [Agravante et al.].

Además de todas las articulaciones que hacen posible el movimiento, el robot cuenta con dos ordenadores integrados. El primero es usado como ordenador de control, encargado de gestionar el robot en sí mismo (motores, navegación, etc), mientras que el otro se utiliza puramente para el apartado multimedia. Relacionado con este apartado multimedia, en el pecho del robot se encuentra una pantalla táctil de 12 pulgadas con una resolución de 1024x768. El uso de esta pantalla cobra relevancia en el proyecto, ya que se utilizará una interfaz gráfica para transmitir comandos al robot y recibir *feedback*.

El sistema operativo utilizado para este robot es Linux Ubuntu 12.04 LTS acompañado del *framework* robótico ROS, del que se hablará detalladamente en el siguiente apartado.

La utilización que se ha dado a este robot hasta la fecha es puramente en el sector de servicios, como azafata para convenciones, conferencias, uso en aeropuertos, guía en museos, etc.[5] Incluso se ha llegado a utilizar como vigilante para detección de intrusos.

En el aspecto audivisual cuenta con 2 cámaras delanteras y una trasera, altavoces, micrófono, pantalla táctil y dos juegos de 8 leds en las orejas que sirven para poder proporcionar todo tipo de contenido al usuario, así como que el usuario sea capaz de comunicarse con el robot a través del micrófono y la pantalla táctil.

[5] https://youtu.be/jVRn9qf6fhM Accedido por última vez el 26/01/2016

2.3.2. Framework robótico ROS

ROS[6] (*Robot Operating System*) es un *framework* utilizado para el desarrollo de software para robots. Desarrollado en 2007 bajo el nombre *Switchyard* por el Laboratorio de Inteligencia Artificial de Stanford como parte de su programa STAIR (*STanford Artificial Intelligence Robot*), proyecto que se encarga del desarrollo de robots capaces de navegar en entornos de oficina, así como interactuar con objetos. Desde ese momento el *framework* crece hasta liberarse la versión ROS Indigo, primera versión LTS (*Long Term Support*) del *framework*, que sigue vigente a fecha del escrito de este documento.

El *framework* provee una colección de herramientas y librerias con el objetivo de simplificar la tarea de crear comportamientos complejos para múltiples plataformas robóticas.

ROS se compone principalmente de dos secciones:

- **ros:** sección del sistema que hace las veces de sistema operativo, con las funciones principales que provee un S.O. tales como paso de mensajes entre procesos, abstracción de hardware o control de dispositivos de bajo nivel entre otros. Su arquitectura está basada en un sistema de grafos, donde cada nodo es capaz de recibir y mandar mensajes de sensores, control, planificación, etc. Todo el código relacionado con esta sección es libre, aplicándose sobre él una licencia de software de tipo BSD (licencia poco restrictiva muy cercana al dominio público que permite libertad tanto para uso comercial como para investigación).
- **ros-pkg:** conjunto de paquetes formado por las aportaciones de los usuarios, que agregan todo tipo de funcionalidades al *framework*. Al ser contribuciones de los usuarios no se rigen por una licencia concreta, estando cubiertos bajo infinidad de licencias distintas en función del paquete utilizado. Entre las principales aplicaciones que podemos encontrar se encuentran sistema de percepción, identificación de objetos, reconocimiento facial, movimiento o planificación.

[6] http://www.ros.org/ Accedido por última vez el 27/01/2016

Actualmente se cuenta con mas de 2000 librerías de software disponible que gestionan todo tipo de funcionalidades en diferentes robots. La lista de robots capaces de funcionar bajo este sistema abarca desde robots humanoides (entre ellos REEM) hasta brazos robóticos.

El uso de este *framework* está ampliamente extendido en el sector de la robótica, pudiendo encontrar multitud de ejemplos de su uso, tales como el desarrollo de una plataforma de control remoto del famoso robot aspirador Roomba [Ruiz et al.], en el que mediante el uso de vectores de velocidad y datos sobre referencias espaciales recogidos del sensor de una Kinect es capaz de controlar el movimiento del robot.

Otro buen ejemplo del uso de ROS y del uso de la interacción humano-robot se encuentra en el desarrollo realizado por el laboratorio PERCRO en Italia [Peppoloni et al. 2015]. En su desarrollo se propone el uso de Leap Motion (controlador que permite el reconocimiento de gestos de las manos) y el uso de realidad virtual para teleoperar un robot. En su propuesta el reconocimiento de gestos de Leap Motion y con la ayuda de una Kinect se capta el movimiento de la persona, que gracias a un casco de realidad virtual es capaz de ver una simulación de la tarea a realizar por el robot. El uso de ROS permite la comunicación con el robot y la posibilidad de realizar la tarea que el usuario esté simulando. En las pruebas realizadas se consiguieron realizar tareas de agarre de objetos con un brazo robótico (Kuka Youbot) con éxito, aunque mencionan que la destreza del robot teleoperado influye positivamente en sus resultados cuanto más hábil sea el robot.

El robot REEM se controla mediante ROS, por lo que su uso es fundamental en este proyecto. Su integración en NAOTherapist se explica en el Capítulo 4.

2.3.3. Framework robótico RoboComp

RoboComp [Manso et al. 2010] es un *framework* robótico que ha sido utilizado tanto por THERAPIST como por NAOTherapist en sus desarrollos, con el objetivo de crear una arquitectura generalizable y reutilizable. RoboComp consta de una serie de componentes de software individuales, pensados para ser distribuidos en diferentes máquinas. Para que se pueda establecer una comunicación entre componentes es necesario que cada componente implemente una interfaz Ice [7] (*Internet Communications Engine*) que facilita a cada componente una comunicación basada en eventos mediante TCP/IP. El hecho de que los componentes se puedan distribuir en diferentes máquinas ofrece una gran ventaja, ya que ciertos componentes pueden funcionar mejor bajo Windows (como la cámara Kinect de Microsoft) mientras que otros pueden tener mayor rendimiento en sistemas UNIX (como los planificadores), además de no ser dependientes de un único lenguaje, ya que cada componente puede estar desarrollado en un lenguaje diferente. Para la realización de este proyecto se utilizarán todos los componentes que conforman la arquitectura NAOTherapist, sustituyendo la interfaz del robot por la nueva que se ha desarrollado en este proyecto.

Si se compara RoboComp con otros *framework* robóticos, en este se intenta solucionar el problema de escalabilidad y reusabilidad de los componentes desarrollados. Debido al desarrollo tan específico que existe normalmente en este campo, es muy difícil generalizar un componente para su uso en diferentes arquitecturas, problema que se solventa en gran medida gracias a este *framework*.

Dado que NAOTherapist funciona bajo el *framework* RoboComp y REEM funciona en ROS, ha sido necesario integrar ambos *framework* para que funcionen simultáneamente sin problemas de compatibilidad. En el Capítulo 6 de este documento se puede observar la complejidad que tuvo este proceso debido al tiempo requerido para su finalización, tal como se indica en la Tabla 6.1.

[7] http://www.zeroc.com/overview.html Accedido por última vez el 31/01/2016

2.3.4. Sensor 3D Kinect

Kinect (ver Figura 2.8) es un controlador de juego desarrollado por Microsoft para su consola Xbox 360. Existe una versión más moderna (Kinect 2) para la nueva consola Xbox One, pero como NAOTherapist aún utiliza la versión antigua todas las referencias sobre la misma serán sobre esa primera versión. Desarrollada por Microsoft se lanza como producto comercial a finales de 2010 con el principal objetivo de controlar la consola mediante comandos de voz y reconocimiento de gestos. Inicialmente solo estaba disponible para Xbox, pero más tarde se añadió compatibilidad con sistemas Windows, aunque gracias a diversas competiciones de la comunidad de software libre se tardó muy poco en conseguir con controlador de código abierto para su uso en GNU/Linux.

Figura 2.8: Kinect.

Se incluye en gran variedad de videojuegos para la consola, que van desde la realización de deportes o baile hasta animales que imitan los movimientos de la persona. Pero gracias a su funcionalidad para recoger una imagen del esqueleto humano y captar comandos de voz, sus aplicaciones fuera del ámbito de la consola se han disparado.

Antes de la llegada de Kinect, el precio de los sensores 3D era muy elevado y no existía un buen software para detección de esqueletos, caras, etc. Desde el lanzamiento en Windows por parte de Microsoft, no solo llegó un sensor 3D producido de forma masiva (y por tanto barato) sino que que el *framework* de utilidades que incluía ha resultado extremadamente útil para todo tipo de aplicaciones de visión por ordenador. Gracias a esto, se ha convertido en el estandar de los sensores 3D en el ámbito de la investigación.

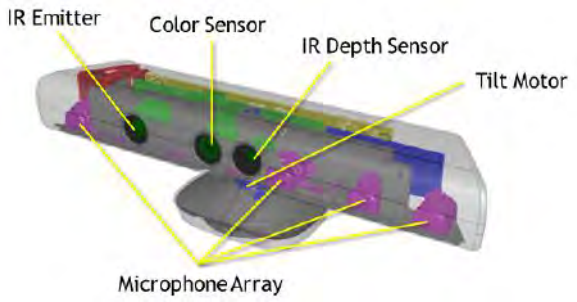

Figura 2.9: Elementos de la Kinect.

El uso de reconocimiento de gestos y posturas es fundamental en el desarrollo de este proyecto, por lo que Kinect es una de las piezas fundamentales del mismo. En la Figura 2.9 se pueden observar los principales elementos que la componen:

- **Cámara RGB:** cámara tradicional que recoge imágenes con una resolución de 1280x960 píxeles. Gracias al sensor de color incorporado es capaz de capturar imágenes en color. Tiene un ángulo de visión de 43º grados en vertical y un campo de vista de 57º horizontal, capaz de capturar 30 imágenes por segundo.
- **Sensores infrarrojos:** el controlador consta de dos sensores infrarrojos. El emisor emite haces de luz mientras que el sensor de profundidad capta los reflejos para obtener la distancia entre el sensor y el objeto, consiguiendo así medir la profundidad de cada parte de la imagen.
- **Micrófono múltiple:** conjunto de cuatro micrófonos que graban audio y la dirección de la onda de sonido.
- **Acelerómetro:** acelerómetro de 3 ejes que consigue obtener la orientación actual de la Kinect.

Entre estas aplicaciones fuera del ámbito propiamente dicho de los juegos, se encuentra su uso como sensor de reconocimiento de objetos, gestos o el esqueleto humano para su aplicación en robótica, planificación de caminos o para el fomento de la interacción humano-robot como ya se ha presentado en la Sección 2.1.1.

Algunos casos de uso recientes que cabe destacar pueden ser el creador de mapas en 2D por el Instituto Tecnológico Sepuluh Nopember de Indonesia [Mardiyanto]. Gracias al uso de Kinect un robot es capaz de crear un mapa en 2D mientras se mueve, dejando constancia de las zonas que ya ha recorrido, pudiendo así volver al punto de inicio en cualquier momento siguiendo dicho mapa. En el ámbito de la teleoperación se encuentran multitud de ejemplos como el ya citado en la Sección 2.3.2. [Ruiz et al.] o el realizado en el Instituto Tecnológico Mangalore en India [Srinivas et al. 2014], que desarrolló un sistema capaz de controlar en tiempo real el movimiento del robot mediante el reconocimiento de gestos de la mano recogidos con Kinect. En sus pruebas, se establecen un número determinado de gestos que indican la dirección que debe seguir el robot.

2.3.5. Simulador Gazebo

Gazebo[8] es un simulador de software libre de robots, con el que se pueden diseñar robots, realizar simulaciones utilizando escenarios realistas tanto interiores como exteriores.

Creado en 2002 por el Dr. Andrew Howard y su estudiante Nate Koenig en la Universidad de California del Sur, surgió a raíz de la necesidad de simular robots en entornos exteriores con diversas condiciones. En 2009 se integró por primera vez junto con ROS, pasando a ser desde ese momento una de sus principales herramientas. En la actualidad, Gazebo sigue en constante desarrollo con ayuda de la comunidad.

El simulador cuenta con acceso a múltiples motores de físicas que determinan como interactúan objetos entre sí al colisionar. También cuenta con gráficos 3D avanzados, que proveen renderizados realistas de entornos, incluyendo luces, sombras y texturas.

Es un simulador completo en lo que a sensores se refiere, capaz de recopilar datos de cámaras, sensores de contacto, etc. gracias a lo cual se pueden realizar simulaciones

[8] http://gazebosim.org/ Accedida por última vez el 31/01/2016

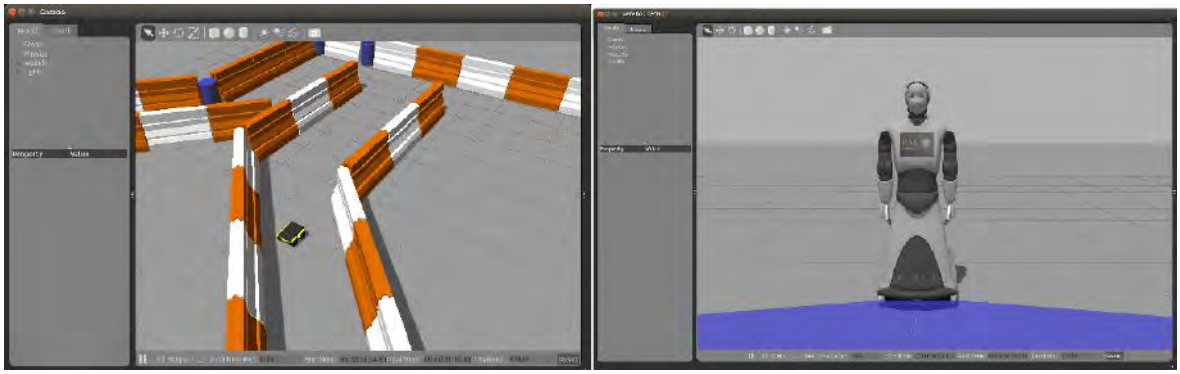

Figura 2.10: Interfaz de Gazebo.

muy precisas del comportamiento del robot. Entre los robots disponibles se encuentran PR2, Pioneer2 DX, REEM o incluso NAO. En cualquier caso, el simulador contiene una herramienta para diseñar un robot propio, por lo que en la práctica este simulador es válido para cualquier robot. Además, al ser software libre, Gazebo cuenta con multitud de plugins externos que hacen de este software un simulador aún más completo y capaz de adaptarse a las necesidades de cada situación.

En la Figura 2.10 se puede ver una muestra de la interfaz del simulador, en la que se observa un ejemplo de entorno sobre el que un robot tiene que planificar un camino.

La utilización de Gazebo ha sido fundamental en el desarrollo de este proyecto, ya que no se ha tenido acceso al robot real para realizar las pruebas, pese a que si ha habido comunicación directa con miembros de la empresa que lo fabrica. Aunque todas las pruebas se han realizado en el simulador, si se han utilizado los sensores reales (Kinect) para la captura de las posturas. Por tanto, para poder ejecutar la arquitectura en el robot físico simplemente habría que conectar el robot a la arquitectura sustituyendo al simulador.

2.4. Retargeting

La versión adaptada de Simon que se utiliza en este proyecto utiliza poses humanas que se han grabado previamente con la Kinect. El robot muestra al jugador dichas poses con sus brazos, por lo que debe imitarlas. Para realizar esto es necesario realizar el *retargeting*, que actualmente funciona con el NAO y que en este proyecto debe funcionar con el REEM. Es necesario observar el funcionamiento de la visión, del esqueleto y del concepto de retargeting para el NAO para poder entender la implementación del Capítulo 4.

Partiendo de un conjunto de ángulos como pueda ser el del esqueleto del cuerpo humano tomado por la Kinect, por ejemplo, realizar un retargeting implicaría recalcular ese conjunto de ángulos según el robot que se esté utilizando para conseguir realizar en él misma postura que refleja el esqueleto. El retargeting que se realiza en este proyecto utiliza el esqueleto recogido de la Kinect para ordenar la ejecución de poses de brazos en el robot REEM. Este sistema de retargeting está basado en el que se utiliza en NAOTherapist, que incluye un sistema de visión y medición de ángulos del esqueleto de Kinect desarrollado por [Rossignoli 2015], y se apoya en algunos conceptos de un trabajo previo de teleoperación de un robot humanoide NAO [Alfaro 2012].

Este sistema recoge el movimiento de brazos y manos a través del sensor de la Kinect para que posteriormente el robot imite la postura. No se contemplan los movimientos de las piernas pese a que el sensor sí los recoge para evitar pérdidas de equilibrio del robot humanoide NAO. El sistema se divide en tres fases que se detallarán a continuación.

2.4.1. Fase 1: Captura de la postura

El sensor Kinect permite recoger capturar información sobre una imagen en tiempo real, de la que se pueden extraer datos como la profundidad de la imagen o captar una serie de puntos característicos del cuerpo humano (ver Figura 2.11). Para captu-

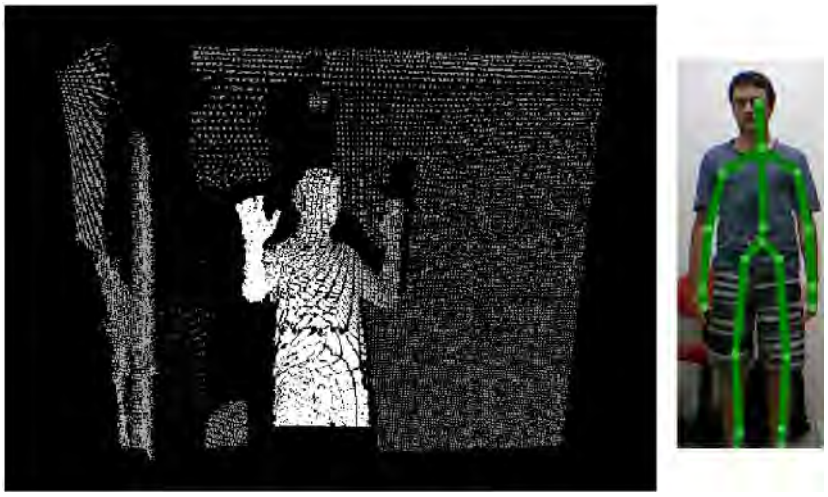

Figura 2.11: Ejemplo de información de profundidad captada por la Kinect y esqueleto humano.

rar la postura se utiliza el componente *Vision* (ver Sección 2.2.1) de la arquitectura NAOTherapist al que se conecta el componente RoboComp *WinKinectComp*. Este componente utliza sotware oficial de Microsoft, por lo que está instalado en un ordenador Windows conectado al sensor Kinect. Proporciona una interfaz que devuelve el esqueleto de la Kinect junto a puntos de la cara en tiempo real. En la Figura 2.12 se puede observar un esqueleto con los principales puntos que se recogen del esqueleto. Una vez recogidos estos datos, la información se trata en el módulo de Visión para obtener conclusiones sobre la postura o el estado del paciente.

2.4.2. Fase 2: Cálculo de los ángulos de las articulaciones

Una vez obtenido el esqueleto, se deben calcular los ángulos de las articulaciones equivalentes a las articulaciones del robot utilizado, para poder realizar una traslación rápida del ángulo que debe tomar el robot. La arquitectura NAOTherapist utiliza como base los planos anatómicos del cuerpo humano. El plano sagital divide el cuerpo en su parte izquierda y derecha, el plano transversal divide el cuerpo por la cintura, teniendo

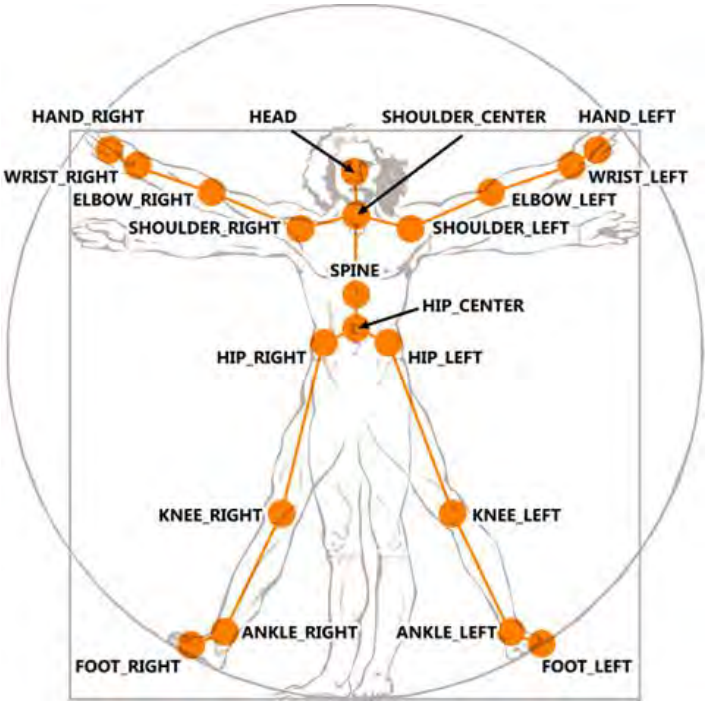

Figura 2.12: Esqueleto humano con los puntos representativos capturados por Kinect.

parte superior y parte inferior, mientras que el plano coronal divide el cuerpo en su parte delantera y trasera, como se puede observar en la Figura 2.13.

A continuación se detalla el ángulo utilizado para cada una de las articulaciones utilizadas basadas en el trabajo de [Rossignoli 2015].

Ángulo de apertura del hombro (ShoulderRoll):

Para el cálculo de la apertura del hombro se utiliza el plano sagital con respecto al brazo, tomando como referencia el seno formado entre ambos vectores.

Ángulo de rotación del hombro (ShoulderPitch):

Para el cálculo de la rotación del hombro se utiliza el ángulo formado entre el plano transversal y el vector correspondiente al brazo, tomando como referencia el seno formado entre ambos vectores.

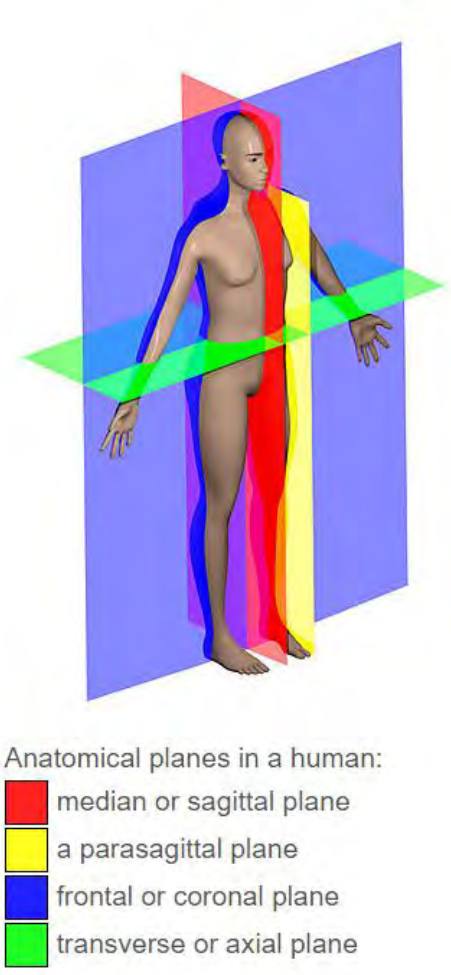

Figura 2.13: Planos anatómicos del cuerpo humano.

Ángulo de apertura del codo (ElbowRoll):

Para establecer el valor de apertura del codo se utiliza el ángulo formado por la propia morfología del codo, del que se toma como referencia el coseno.

Ángulo de rotación del codo (ElbowYaw):

El proceso de cálculo del ángulo de rotación del codo no es tan simple, ya que su posición cambia con el giro del hombro, por lo que debe ser dependiente de la posición del hombro en todo momento. Para ello se distinguen dos casos, dependiendo de si el

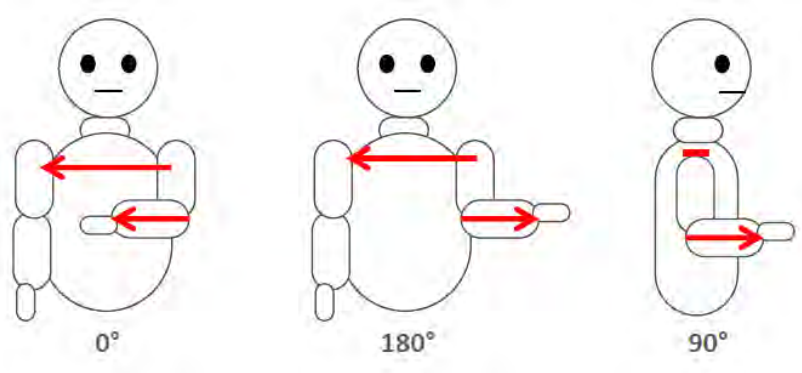

Figura 2.14: Ángulo de rotación del codo con brazo cerrado.

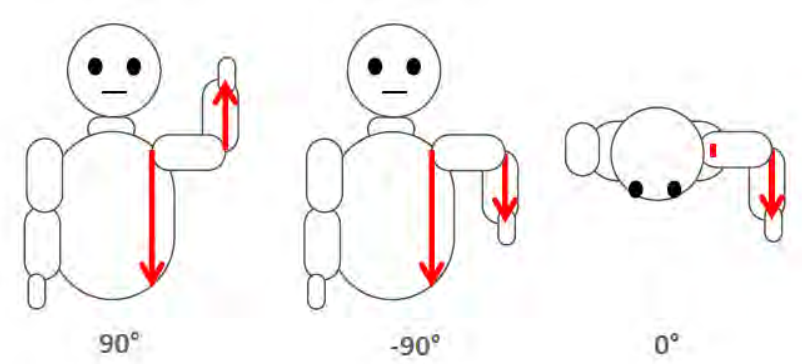

Figura 2.15: Ángulo de rotación del codo con brazo abierto.

brazo está pegado al cuerpo o está extendido. Este cálculo está basado en el trabajo de [Alfaro 2012].

En el caso del brazo pegado al cuerpo, se toma como referencia el vector que une los hombros. Este ángulo toma aquí el valor 0º con el brazo doblado hacia dentro, 90º si se gira hacia el frente, y 180º en el caso de abrirse hacia fuera, como se ve en la Figura 2.14.

En el caso del brazo extendido, se tomará el vector que une el hombro con el lado de la cadera de ese mismo lado. Los valores para este ángulo van desde -90º si se dobla hacia abajo hasta 90º si se dobla hacia arriba, teniendo un valor de 0º con el brazo

doblado hacia el frente (ver Figura 2.15).

Atendiendo a las imágenes se aprecia que no es posible utilizar el mismo vector para ambos casos, ya que si se utiliza el vector que une los hombros con el brazo abierto siempre se obtendría un ángulo de 90º. Para que la transición no sea demasiado brusca y pase de tomar un vector a tomar el otro directamente, se ha planteado la siguiente fórmula:

$$\frac{ac}{\frac{\pi}{2}} + b(1 - \frac{c}{\frac{\pi}{2}})$$

Donde:

- RC = Ángulo de rotación del codo
- a = Ángulo de rotación del codo calculado mediante el vector que une el hombro a la cadera.
- b = Ángulo de rotación del codo calculado mediante el vector que une los hombros.
- c = Ángulo de apertura del hombro.

Con esta fórmula se consigue que al tener una apertura de hombro de 90º se anule el sumando, teniendo solamente en cuenta el ángulo calculado con el vector del hombro a la cadera. A la inversa, si la apertura del hombro son 0º, solo se tendrá en cuenta el primer sumando, utilizando el vector que une los hombros. En el resto de casos intermedios, se utilizará una ponderación dando más peso al primer o segundo sumando dependiendo de la apertura del hombro. Una aplicación real de este cálculo puede verse en la Figura 2.16.

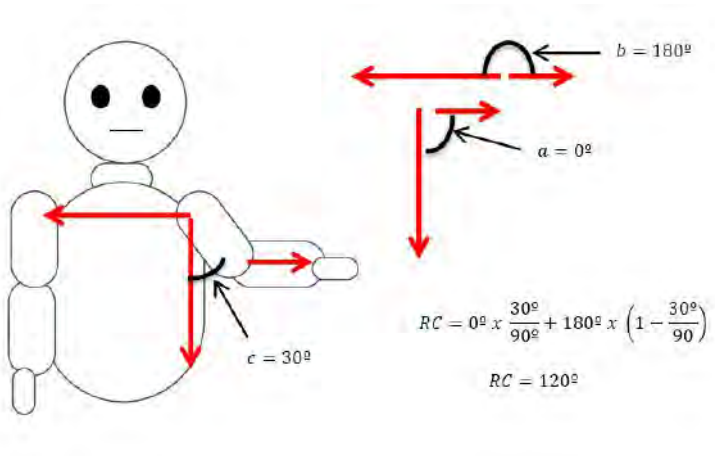

Figura 2.16: Ejemplo de cálculo de la rotación del codo.

2.4.3. Retargeting original del robot NAO

Una vez realizados los cálculos de los ángulos de las extremidades superiores es necesario realizar ciertas modificaciones en los mismos. Estas modificaciones pueden consistir en escalar o invertir algunos ángulos, debido a que el robot utilizado no utilice exactamente los mismos planos que se han usado al calcular los ángulos.

El retargeting se implementó para el NAO en el proyecto de [Alfaro 2012] y se adaptó en NAOTherapist. Uno de estos cambios por ejemplo se encuentra en la apertura del hombro. Para el modelo planteado, la apertura es 90º cuando el brazo está abierto. En cambio, en el NAO este valor es 90º en caso de ser el brazo izquierdo, pero -90º si es el derecho, por lo que es necesario cambiar el signo del brazo derecho para ajustarlo. También hay diferencias con el ángulo de rotación del hombro, que en el modelo va desde 0º (abajo) hasta 180º (arriba), mientras que para el NAO abajo es 90º, 0º es el brazo extendido y -90º es arriba.

El cálculo de todas estas diferencias es lo que se denomina retargeting, en el que se consigue adaptar un modelo general como el explicado a las peculiaridades de un robot concreto como es el NAO. En el desarrollo de este proyecto se realiza dicha labor de retargeting con el robot REEM partiendo del modelo de ángulos proporcionados

por el componente Vision de NAOTherapist y resolviendo las dificultades particulares encontradas en este robot concreto, que se detallará en el capítulo 4 del presente documento. Una vez obtenida la transformación de los ángulos se comunicarán los valores al robot para que adopte la pose requerida.

Capítulo 3

Análisis y diseño del sistema

En este capítulo del documento se analizará y planteará el diseño del sistema desarrollado. Para ello se establecerán las normas y restricciones el proyecto, así como los elementos de software y hardware utilizados. También se especificarán los requisitos que debe cumplir el sistema, y se plantearán distintos casos de uso para que quede perfectamente detallado cómo debe ser el funcionamiento del sistema. La arquitectura del sistema es la que se puede ver en la Figura 3.1, similar a la que se detalló en el capítulo 2 de este documento. La diferencia radica en el componente de la interfaz del robot NAO, que ha sido sustituido por el del robot REEM.

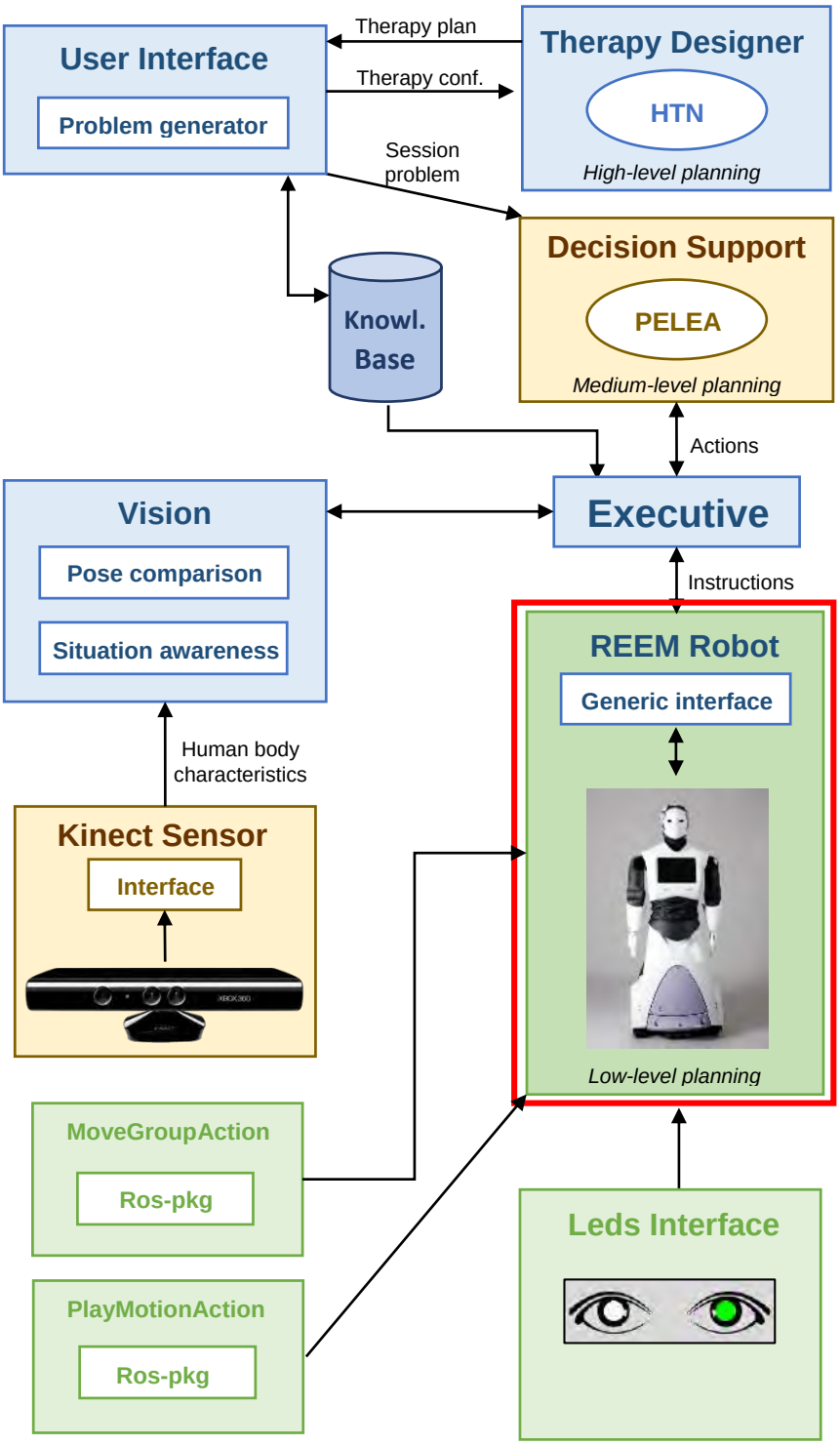

Figura 3.1: Arquitectura del sistema.

3.1. Normas y restricciones del proyecto

Las **normas** que se aplican al proyecto son las siguientes:

- Dado que se utiliza la cámara de la Kinect y esta puede recoger imágenes de los pacientes/usuarios del sistema, es necesario cumplir con la Ley Orgánica 15/1999 de Protección de Datos de Carácter Oficial[1].
- Aunque el objetivo del proyecto no es su aplicación a terapias de rehabilitación como el NAOTherapist original, puede cumplir perfectamente dicha función. Al realizarse dichas terapias con pacientes menores de edad, se debe cumplir la Ley 26/2015, de 28 de julio, de modificación del sistema de protección a la infancia y a la adolescencia[2].

En cuanto a las **restricciones**, se dividen entre restricciones de hardware y restricciones de software. Las restricciones de hardware son las siguientes:

- El controlador para recoger la imagen del usuario debe ser una Kinect en su primer modelo desarrollado para Xbox 360.
- El ordenador en el que se ejecute el sistema debe contener un controlador gráfico Nvidia para el correcto funcionamiento del simulador.
- La Kinect debe estar conectada a un ordenador que hace las veces de servidor y recoge y envía los datos al resto del sistema, que se ejecuta en un ordenador diferente.

[1] http://www.boe.es/buscar/act.php?id=BOE-A-1999-23750 Accedido por última vez el 01/02/2016

[2] https://www.boe.es/diario_boe/txt.php?id=BOE-A-2015-8470 Accedido por última vez el 01/02/2016

Las restricciones de software son las siguientes:

- La versión de ROS utilizada debe ser ROS Hydro.
- Para que la simulación de REEM funcione se debe desplegar el sistema en un ordenador con Ubuntu 12.04.
- Se debe instalar el controlador privativo de Nvidia para que la ejecución de Gazebo sea posible.
- Restricciones propias de la arquitectura del robot. En la sección REEM del capítulo 2 se detallan los valores máximos y mínimos que puede alcanzar cada articulación.

Por último mencionar que tanto ROS como RoboComp son de libre acceso, mientras que la arquitectura NAOTherapist y el componente desarrollado para ésta pertenecen a la Universidad Carlos III de Madrid, siendo por tanto de uso restringido. Por otra parte, todas las herramientas utilizadas del robot REEM son de libre acceso, pero no se puede garantizar que si la empresa PAL Robotics añade o modifica herramientas éstas sigan siendo de libre acceso.

3.2. Entorno operacional

En este apartado se detallan las herramientas tanto a nivel de hardware como de software utilizadas para el desarrollo del proyecto:

- Ordenador Mountain F-13 con procesador Intel Core I7-4700MQ, 8GB de RAM, Nvidia GTX 765M y sistema operativo Ubuntu 12.04-Desktop x64.
- Cámara Kinect Xbox 360 conectada a PC con Windows 7 Pro-N x64. Intel Core2-Quad Q9200, 4GB de Ram y Nvidia GForce 840GS.

- Simulador Gazebo junto con el paquete del robot REEM.
- Simulador Choregraphe para robot NAO.
- Robot NAO.
- Editor de textos Sublime Text para desarrollo en Python.
- Latex, con el editor online ShareLateX[3].
- Gantt project.
- Adobe Acrobat XI Pro.
- Paquete ofimático de Microsoft Office 2013.

3.3. Especificación de requisitos

En el siguiente apartado se establecerá la definición de los requisitos que debe cumplir el proyecto que se va a desarrollar. Para ello se elaborará una tabla por cada requisito especificado, que tendrá el formato que se puede ver en la Tabla 3.1.

RX-Y	
DESCRIPCIÓN:	
PRIORIDAD: ☐ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☐ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☐ Alta ☐ Media ☐ Baja

Tabla 3.1: Modelo de tabla para la especificación de requisitos.

[3]https://es.sharelatex.com Accedido por última vez el 01/02/2016

Donde:

- **RX-Y:** identificador único del requisito, siendo:
  - R: Valor fijo que indica que se trata de un requisito.
  - X: Valor que especifica si el requisito es funcional "F" o no funcional "NF".
  - Y: Identificador numérico del requisito. Es un valor entero.
- **Descripción:** descripción clara y concisa del requisito.
- **Prioridad:** cada requisito incluye una medida de la prioridad que posee, con el fin de que el desarrollador pueda decidir la planificación del desarrollo.
- **Necesidad:** representa la importancia que toma el requisito para el desarrollo del proyecto. Puede tomar tres valores: *esencial*, requisito imprescindible para poder llevar a cabo el proyecto; *deseable*, requisito que se desea cumplir pero no se puede garantizar su cumplimentación; y *opcional*, requisito que puede o no ser cumplido y no impide que el proyecto pueda ser finalizado.
- **Claridad:** un requisito es claro si tiene una única interpretación, por lo que tomará el valor *Alta*. Si da lugar a pequeñas ambigüedades, su claridad será *Media*. Por último, un requisito totalmente ambiguo tendrá una claridad *Baja*.
- **Verificabilidad:** un requisito es verificable si se puede comprobar que dicho requisito se ha incorporado en el diseño del proyecto. Si un requisito es fácilmente verificable, tomará el valor *Alta*. Si requiere de un pequeño estudio para comprobar que se esté cumpliendo su verificabilidad será *Media*. Por último, si el requisito no se puede comprobar o no depende estrictamente del sistema su comprobación, su verificabilidad será *Baja*.

Se distinguirá entre **requisitos funcionales**, aquellos que indican qué debe hacer el software en cuestión, y **requisitos no funcionales**, que son aquellos que imponen

restricciones en el diseño o la implementación. Es decir, son propiedades o cualidades que el producto debe tener. Destacar que al hablar de "El sistema" en los requisitos se hace referencia a "Arquitectura NAOTherapist basada en el *framework* robótico RoboComp".

3.3.1. Requisitos funcionales

RF-01	
DESCRIPCIÓN: Si el sistema está en modo espejo, el robot deberá realizar la misma posición que se reciba del esqueleto de la Kinect.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.2: Requisito funcional 01.

RF-02	
DESCRIPCIÓN: El robot podrá generar una reproducción hablada a partir de un texto (*text-to-Speech*).	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☐ Alta ☒ Media ☐ Baja

Tabla 3.3: Requisito funcional 02.

RF-03	
DESCRIPCIÓN: El robot podrá reproducir ficheros de audio.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☐ Alta ☒ Media ☐ Baja

Tabla 3.4: Requisito funcional 03.

RF-04	
DESCRIPCIÓN: El robot podrá realizar animaciones con sus articulaciones.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.5: Requisito funcional 04.

RF-05	
DESCRIPCIÓN: El sistema realizará una adaptación de los ángulos del esqueleto de la Kinect a los ángulos del robot.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.6: Requisito funcional 05.

RF-06	
DESCRIPCIÓN: El sistema incluirá una interfaz que simule los botones y leds físicos de los ojos del robot REEM.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.7: Requisito funcional 06.

RF-07	
DESCRIPCIÓN: La ejecución de Simon podrá tener distintos niveles de dificultad con distinto número de posiciones disponibles.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☐ Alta ☒ Media ☐ Baja

Tabla 3.8: Requisito funcional 07.

RF-08	
DESCRIPCIÓN: La reproducción de sonido podrá ser o no bloqueante. Es bloqueante cuando el programa no prosigue su ejecución hasta que el audio no ha terminado, y no bloqueante cuando el audio se reproduce en segundo plano.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☐ Alta ☒ Media ☐ Baja	**VERIFICABILIDAD:** ☐ Alta ☒ Media ☐ Baja

Tabla 3.9: Requisito funcional 08.

RF-09	
DESCRIPCIÓN: La interfaz dispondrá de 6 botones que simulan los botones físicos del robot NAO.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.10: Requisito funcional 09.

RF-10	
DESCRIPCIÓN: La interfaz podrá ser personalizable con diferentes aspectos visuales como los detallados en el Capítulo 4 del documento.	
PRIORIDAD: ☐ Alta ☐ Media ☒ Baja	**NECESIDAD:** ☐ Esencial ☐ Deseable ☒ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.11: Requisito funcional 10.

RF-11	
DESCRIPCIÓN: Se deberá crear un manual de usuario que explique cómo instalar y ejecutar el sistema.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.12: Requisito funcional 11.

RF-12	
DESCRIPCIÓN: Si no se realiza la conexión con Gazebo se mostrará el correspondiente mensaje de error.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.13: Requisito funcional 12.

RF-13	
DESCRIPCIÓN: Si no se realiza la conexión con ROS se mostrará el correspondiente mensaje de error.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.14: Requisito funcional 13.

RF-14	
DESCRIPCIÓN: El formato de los ficheros de audio debe ser *.wav*.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.15: Requisito funcional 14.

3.3.2. Requisitos no funcionales

RNF-01	
DESCRIPCIÓN: La arquitectura NAOTherapist, basada en el framework robótico RoboComp, debe conectar sin incompatibilidades con la parte del framework robótico ROS que controla el robot REEM.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.16: Requisito no funcional 01.

RNF-02	
DESCRIPCIÓN: El sistema debe conectar correctamente con el simulador Gazebo	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.17: Requisito no funcional 02.

RNF-03	
DESCRIPCIÓN: Para que el simulador Gazebo conecte debe estar inicializado ROS mediante el comando *roscore.*	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.18: Requisito no funcional 03.

RNF-04	
DESCRIPCIÓN: El sistema debe ejecutarse en un equipo que tenga un controlador gráfico Nvidia.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.19: Requisito no funcional 04.

RNF-05	
DESCRIPCIÓN: La interfaz debe ejecutarse en un hilo de ejecución diferente al del programa principal.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.20: Requisito no funcional 05.

RNF-06	
DESCRIPCIÓN: A excepción del componente *Therapy Designer* todos los demás componentes del sistema deben estar en ejecución para poder recibir datos de la Kinect.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.21: Requisito no funcional 06.

RNF-07	
DESCRIPCIÓN: Se debe utilizar la versión de ROS Hydro.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.22: Requisito no funcional 07.

RNF-08	
DESCRIPCIÓN: El sistema se debe ejecutar sobre el sistema operativo Ubuntu 12.04.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.23: Requisito no funcional 08.

RNF-09	
DESCRIPCIÓN: La interfaz debe poder ejecutarse sobre la interfaz de Ubuntu Openbox.	
PRIORIDAD:	**NECESIDAD:**
☒ Alta	☐ Esencial
☐ Media	☒ Deseable
☐ Baja	☐ Opcional
CLARIDAD:	**VERIFICABILIDAD:**
☒ Alta	☒ Alta
☐ Media	☐ Media
☐ Baja	☐ Baja

Tabla 3.24: Requisito no funcional 09.

RNF-10	
DESCRIPCIÓN: No se podrán producir colisiones en el movimiento del robot.	
PRIORIDAD:	**NECESIDAD:**
☒ Alta	☒ Esencial
☐ Media	☐ Deseable
☐ Baja	☐ Opcional
CLARIDAD:	**VERIFICABILIDAD:**
☐ Alta	☐ Alta
☒ Media	☒ Media
☐ Baja	☐ Baja

Tabla 3.25: Requisito no funcional 10.

RNF-11	
DESCRIPCIÓN: Las imágenes utilizadas en la interfaz deben tener formato *.gif*.	
PRIORIDAD:	**NECESIDAD:**
☒ Alta	☒ Esencial
☐ Media	☐ Deseable
☐ Baja	☐ Opcional
CLARIDAD:	**VERIFICABILIDAD:**
☒ Alta	☒ Alta
☐ Media	☐ Media
☐ Baja	☐ Baja

Tabla 3.26: Requisito no funcional 11.

RNF-12	
DESCRIPCIÓN: El tamaño de la ventana de la interfaz debe ser de 1024x768.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.27: Requisito no funcional 12.

RNF-13	
DESCRIPCIÓN: El sistema se desarrollará en el lenguaje de programación Python.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.28: Requisito no funcional 13.

RNF-14	
DESCRIPCIÓN: El componente desarrollado ReemComp deberá funcionar completamente integrado en la arquitectura NAOTherapist.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☐ Alta ☒ Media ☐ Baja

Tabla 3.29: Requisito no funcional 14.

RF-15	
DESCRIPCIÓN: El color verde en la interfaz indicará que la posición tomada por el usuario es correcta.	
PRIORIDAD:	**NECESIDAD:**
☒ Alta	☒ Esencial
☐ Media	☐ Deseable
☐ Baja	☐ Opcional
CLARIDAD:	**VERIFICABILIDAD:**
☒ Alta	☒ Alta
☐ Media	☐ Media
☐ Baja	☐ Baja

Tabla 3.30: Requisito no funcional 15.

RNF-16	
DESCRIPCIÓN: El color blanco en la interfaz indicará que la posición tomada por el usuario es incorrecta.	
PRIORIDAD:	**NECESIDAD:**
☒ Alta	☒ Esencial
☐ Media	☐ Deseable
☐ Baja	☐ Opcional
CLARIDAD:	**VERIFICABILIDAD:**
☒ Alta	☒ Alta
☐ Media	☐ Media
☐ Baja	☐ Baja

Tabla 3.31: Requisito no funcional 16.

RNF-17	
DESCRIPCIÓN: Las articulaciones del robot solo podrán tomar un valor entre el máximo y el mínimo especificados según las características del robot.	
PRIORIDAD:	**NECESIDAD:**
☒ Alta	☒ Esencial
☐ Media	☐ Deseable
☐ Baja	☐ Opcional
CLARIDAD:	**VERIFICABILIDAD:**
☒ Alta	☒ Alta
☐ Media	☐ Media
☐ Baja	☐ Baja

Tabla 3.32: Requisito no funcional 17.

RNF-18	
DESCRIPCIÓN: En modo espejo, el robot deberá imitar la postura del usuario en menos de 3 segundos.	
PRIORIDAD: ☐ Alta ☒ Media ☐ Baja	**NECESIDAD:** ☐ Esencial ☒ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.33: Requisito no funcional 18.

RNF-19	
DESCRIPCIÓN: Las secuencias del juego Simon deberán tener como mínimo 3 posturas diferentes que el usuario deberá repetir.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.34: Requisito no funcional 19.

RNF-20	
DESCRIPCIÓN: La interfaz debe iniciarse antes del inicio de la simulación de Simon.	
PRIORIDAD: ☒ Alta ☐ Media ☐ Baja	**NECESIDAD:** ☒ Esencial ☐ Deseable ☐ Opcional
CLARIDAD: ☒ Alta ☐ Media ☐ Baja	**VERIFICABILIDAD:** ☒ Alta ☐ Media ☐ Baja

Tabla 3.35: Requisito no funcional 20.

3.4. Especificación de casos de uso

En este punto se detallan los casos de uso asociados al sistema. Un caso de uso es una descripción de los pasos que se deben seguir para la realización de una tarea concreta.

3.4.1. Descripción tabular de casos de uso

Los casos de uso se presentarán en forma de tabla, siguiendo el formato de la Tabla 3.36.

CU-X
NOMBRE:
ACTOR:
DESCRIPCIÓN:
PRECONDICIONES:
POSTCONDICIONES:

Tabla 3.36: Modelo de tabla para la definición de casos de uso.

Donde:

- **CU-X:** identificador único del requisito, siendo:
  - CU: Caso de Uso.
  - X: Identificador numérico del requisito. Es un valor entero.
- **Nombre:** nombre del caso de uso.
- **Actor:** tipo de usuario que se encarga de realizar el caso de uso.
- **Descripción:** especifica qué debe realizar el caso de uso.
- **Precondiciones:** conjunto de condiciones que se deben cumplir para poder realizar el caso de uso.

- **Postcondiciones:** estado en el que queda el sistema una vez realizado el caso de uso.

CU-01
NOMBRE: Detección del usuario
ACTOR: Desarrollador/usuario
DESCRIPCIÓN: Comprobar que la arquitectura está reconociendo correctamente al usuario.
PRECONDICIONES: • La arquitectura NAOTherapist debe estar en ejecución. • El sensor Kinect debe estar conectado. • El usuario debe estar situado enfrente de la Kinect.
POSTCONDICIONES: • En caso de éxito, el sistema sigue con su ejecución, indicando mediante voz al usuario el siguiente paso. • En caso de no detectar al usuario, el robot pregunta al usuario "¿Dónde estás?"

Tabla 3.37: Caso de uso 01.

CU-02
NOMBRE: Ejecución de una postura.
ACTOR: Desarrollador/usuario
DESCRIPCIÓN: El robot adopta una postura que el usuario debe imitar y comprueba que se haya realizado correctamente.
PRECONDICIONES: • El sistema debe estar en ejecución, con ROS y Gazebo lanzados y en espera. • El robot debe indicar una postura. • La interfaz debe estar en ejecución. • El usuario debe estar enfrente de la Kinect.
POSTCONDICIONES: • En caso de éxito, los ojos de la interfaz se ponen en verde y se emite un sonido que indica que la postura ha sido correcta. • En caso de que la postura no sea correcta, los ojos de la interfaz se muestran en blanco, y el robot intenta corregir la postura del usuario.

Tabla 3.38: Caso de uso 02.

CU-03
NOMBRE: Descansar.
ACTOR: Desarrollador/usuario
DESCRIPCIÓN: El robot hace una pausa, realizando una animación predefinida para que el usuario relaje los brazos.
PRECONDICIONES: • El sistema debe estar en ejecución, con ROS y Gazebo lanzados y en espera. • El usuario debe estar enfrente de la Kinect.
POSTCONDICIONES: • El robot realiza una animación predefinida como inspirar/expirar o bailar la macarena.

Tabla 3.39: Caso de uso 03.

3.4.2. Descripción gráfica de casos de uso

En la Figura 3.2 se puede ver un diagrama de los casos de uso.

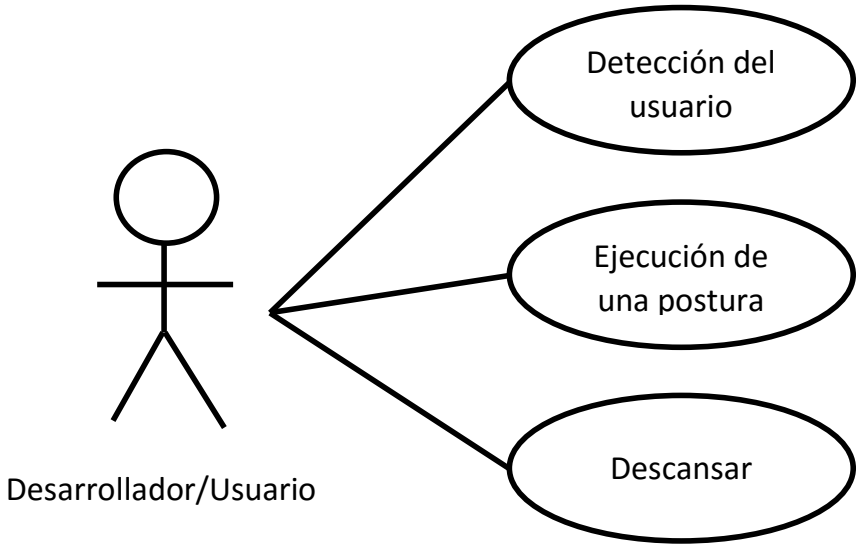

Figura 3.2: Diagrama de casos de uso.

3.5. Metodología

La metodología utilizada durante el desarrollo de esta arquitectura es una metodología basada en **prototipos**. Destacar que esta metodología es exclusiva de la implementación de la arquitectura, mientras que la metodología en cascada a la que se hace referencia en el Capítulo 6 es sobre la planificación general del proyecto. Este tipo de de modelos de desarrollo evolutivo permiten construir rápidamente ciertos componentes del sistema para que el desarrollador pueda comprender su funcionamiento, así como las necesidades del cliente. Gracias a este sistema se consigue una versión temprana que da una idea clara de lo que se necesita o las posibilidades que ofrece el sistema, permitiendo refinar los requisitos iniciales y los métodos de desarrollo utilizados para su implementación. En la Figura 3.3 se puede observar el ciclo de desarrollo siguiendo esta metodología.

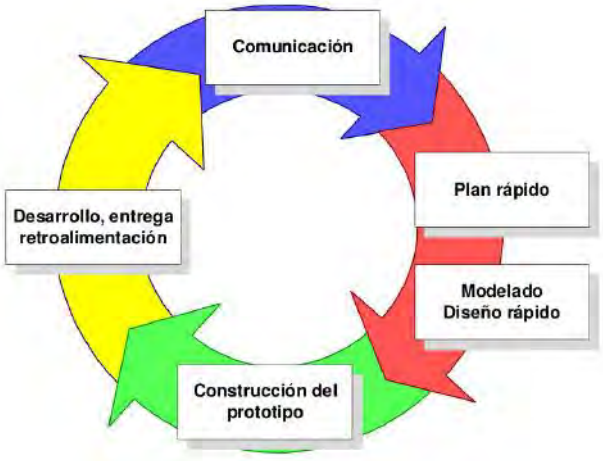

Figura 3.3: Metodología basada en prototipos.

En el caso del proyecto de fin de grado, se decidió el uso de este tipo de metodología ya que se tenía una idea bastante clara de lo que se necesitaba (puesto que se parte de la base de NAOTherapist), pero no se conocía la mejor manera de implementarlo debido a las diferencias con el robot NAO y las peculiaridades propias del sistema.

Capítulo 4

Implementación del sistema

En este capítulo se detalla todo el proceso de implementación del sistema, partiendo de la integración de las tecnologías necesarias para su desarrollo, así como los métodos y clases programadas. Por último se explicará la integración del entorno de pruebas del juego Simon, que será objeto de la evaluación del sistema en el siguiente capítulo del presente documento.

4.1. Integración de los componentes

Como ya se ha mencionado en capítulos anteriores, el objetivo de este proyecto es crear un nuevo componente basado en el robot REEM para la arquitectura NAOTherapist [González et al. 2015]. Como se puede ver en la Figura 3.1, la idea de este proyecto es mantener la arquitectura original, sustituyendo completamente el componente del robot NAO por otro adaptado al REEM. Por ello, es necesario instalar toda la arquitectura NAOTherapist e integrar el componente desarrollado para REEM dentro del *framework* robótico RoboComp para asegurar la compatibilidad con el resto de la misma.

4.1.1. Integración NAOTherapist original

En primer lugar se ha instalado la arquitectura NAOTherapist. Para ello se han seguido los pasos detallados en la Sección 9.2 de este mismo documento. Respecto a esta parte no se ha tenido ningún problema a la hora de su integración, puesto que esta arquitectura funciona igualmente bajo Ubuntu 12.04 y no se ha precisado ninguna corrección o añadido adicional para tener este sistema en funcionamiento. El componente del robot NAO que se desea cambiar funciona sobre el robot real. En el laboratorio se encuentra disponible un robot NAO, por lo que se ha podido probar el correcto funcionamiento de la arquitectura tanto con el robot físico como con una versión de evaluación de Choregraphe, simulador del robot NAO.

Inicialmente se ha realizado una ejecución completa de la arquitectura NAOTherapist original con el robot NAO real haciendo uso de un script *light.sh* que lanza todos los componentes necesarios.

En un primer momento se pensó en expandir el componente NAOComp para que soportase el robot REEM, pero se tuvo que descartar debido a que es un componente muy dependiente del robot, por lo que se hizo más factible realizar un componente nuevo específico de REEM. Por tanto, se ha tenido que editar el script *light.sh* para que no inicie la planificación de alto nivel, es decir, para que no utilice el componente *NAOComp* y arranque el nuevo componente a desarrollar, *ReemComp*.

Para conectar con el resto de la arquitectura se requería que el nuevo componente formase parte del *framework* RoboComp. Las funciones que se implementan en NAOComp están definidas en la interfaz Ice de RobotControl. Esta interfaz del componente es utilizada para comunicarse con el resto de la arquitectura, concretamente con el Ejecutivo. El objetivo de este trabajo es mantener esta interfaz, recreando en el componente *ReemComp* todas las funciones definidas. Esto asegura que la compatibilidad con la arquitectura será completa.

4.1.2. Integración ROS

Se ha estudiado el funcionamiento del robot REEM y su forma de uso bajo un simulador, para lo que ha sido necesaria la integración del *framework* ROS para las comunicaciones entre el componente y el robot. Esta tecnología permite enviar ordenes al robot a través de código desarrollado en Python. Inicialmente se pensó en programar todo en C++ ya que es más rápido, pero dado que el componente no tenía una gran carga de procesamiento y la mayoría de componentes de la arquitectura estaban en Python, se optó por este último manteniendo la homogeneidad de la arquitectura en la medida de lo posible. ROS permite comunicarse con el robot para realizar las diferentes posturas y movimientos del robot, además de las animaciones predefinidas. Estas animaciones, que se detallan en la Sección 4.3, son movimientos complejos como un saludo, un baile, etc.

Como no se tenía experiencia con esta tecnología a la hora de empezar el proyecto, primeramente se realizaron una serie de tutoriales que provee la página oficial de ROS[1]. En estos tutoriales se explica el funcionamiento básico del sistema, por lo que en un inicio fueron de gran utilidad para comprender el funcionamiento de este conjunto de herramientas.

Integrar ROS junto con RoboComp era un requisito indispensable para este proyecto. Fue la primera parte del trabajo y consumió bastante tiempo del desarrollo, tal como puede observarse en el diagrama de Gantt de la Sección 6.1 de planificación del desarrollo. Ha sido necesario contactar directamente con los ingenieros de la empresa PAL Robotics para poder solucionar algunos problemas asociados al robot REEM.

El resultado de esta investigación para instalar la versión de ROS requerida por REEM sin que sea incompatible con RoboComp (requerido por NAOTherapist) es un manual que se detalla en la Sección A.1.1 del **Apéndice A: Manual de instalación**. De este estudio se ha obtenido conocimiento del arranque del *framework* y su puesta en ejecución, de la creación de paquetes o del paso de mensajes entre dos componentes,

[1] http://wiki.ros.org/ROS/Tutorials Accedido por última vez el 07/02/2016

elemento fundamental para comunicar al robot las ordenes pertinentes. Este punto ha resultado complicado puesto que no se había tratado hasta la fecha con ningún *framework* robótico similar.

4.1.3. Integración Gazebo

Dado que el robot REEM físico está en la sede de PAL Robotics en Cataluña, no teníamos acceso directo a él (aunque hay planes de futuro para poder probar este proyecto en su robot, ya que tras haber contactado con ellos en varias ocasiones han mostrado interés por el resultado final del proyecto). PAL Robotics provee un modelo del robot REEM para el simulador ROS Gazebo, que es lo que se ha utilizado en este proyecto.

Por tanto, es necesaria la instalación del simulador Gazebo junto con el paquete de REEM para poder visualizar el robot en dicho simulador. Los pasos de instalación se encuentran detallados en el **Apéndice A: Manual de instalación**. REEM funciona bajo ROS, por lo que la misma página oficial de ROS facilita la instalación tanto de Gazebo como del paquete que contiene todo lo necesario para la ejecución de REEM[2]. En este punto se han tenido problemas durante su integración debido a un problema con el propio código del robot REEM que impedía su correcta instalación. Afortunadamente, en este trabajo se tuvo contacto directo con los ingenieros de PAL Robotics, que actualizaron rápidamente su repositorio de código gracias al feedback que se les proporcionó. También fue necesaria la instalación de controladores privativos especiales del controlador gráfico Nvidia, ya que Gazebo no es capaz de funcionar en caso de no tenerlos.

Tras solucionar todos los problemas técnicos para instalar RoboComp y ROS simultáneamente, se logró tener la plataforma lista para poder empezar a investigar cómo desarrollar el nuevo componente. En la Figura 4.1 se puede observar cómo Gazebo representa al robot REEM una vez que se carga todo.

[2]http://wiki.ros.org/Robots/REEM Accedido por última vez el 07/02/2016

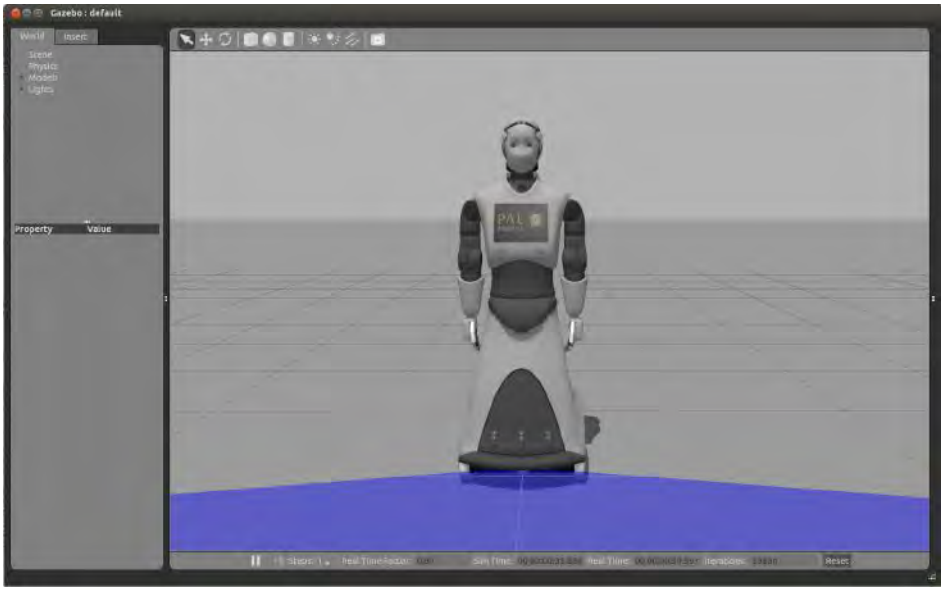

Figura 4.1: Interfaz de Gazebo con REEM.

4.1.4. Análisis del robot REEM

Antes de empezar con la integración del nuevo componente en la arquitectura NAOTherapist, se ha realizado un profundo estudio de todo el robot REEM, examinando las funcionalidades incluidas en su paquete. De este estudio se han obtenido varios elementos fundamentales para el desarrollo del proyecto. El primero de ellos es la herramienta **MoveIt Setup Assistant**, que muestra gráficamente en detalle todas las articulaciones del robot y el movimiento de cada una de ellas (ver Figura 4.4). Esto permitió conocer la morfología del robot y ver cómo realizaba cada movimiento, por lo que se pudo obtener información suficiente para hacer el retargeting del esqueleto de la Kinect al robot REEM, tomando como base la idea que ya estaba programada para el NAO, detallada en la Sección 4.3 de este mismo capítulo.

La empresa PAL Robotics ha facilitado la herramienta de creación de animaciones, basada en ROS, *rqt-joint-trajectory-controller* (ver Figura 4.2). Esta herramienta, similar a MoveIt Setup Assistant, despliega un listado de las articulaciones disponibles del robot, donde se puede modificar su valor. A diferencia de MoveIt Setup Assistant,

Figura 4.2: Herramienta para la creación de animaciones.

que tiene un propio modelo del robot REEM, esta Funciona a través de Gazebo, y permite indicar una posición concreta al robot para obtener los ángulos de dicha posición e incluirlos en el fichero de animaciones.

Dentro del paquete de REEM que se ha instalado junto con Gazebo se ofrece un conjunto de herramientas y utilidades del robot, entre los cuales se encuentra *PlayMotion*, que ejecuta una animación predefinida o una postura simple desde una terminal o a través de una interfaz gráfica (ver Figura 4.3). Gracias a este estudio también se ha obtenido la forma de enviar órdenes al robot a través de código desarrollado en Python, elemento clave para el componente que se quiere desarrollar y que se explicará en la Sección 4.1.5.

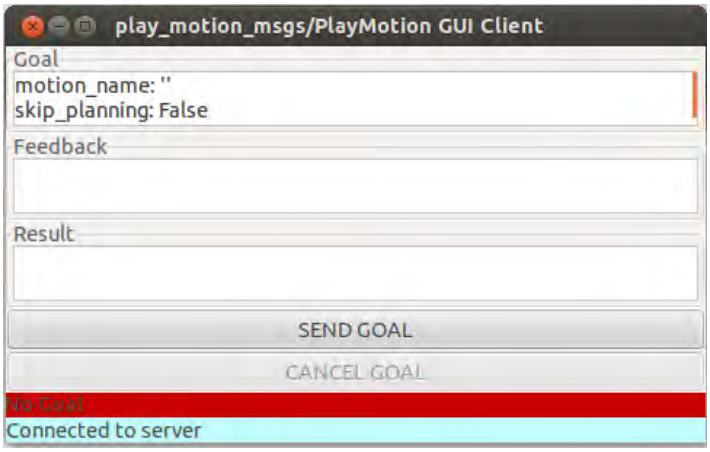

Figura 4.3: Interfaz de PlayMotion.

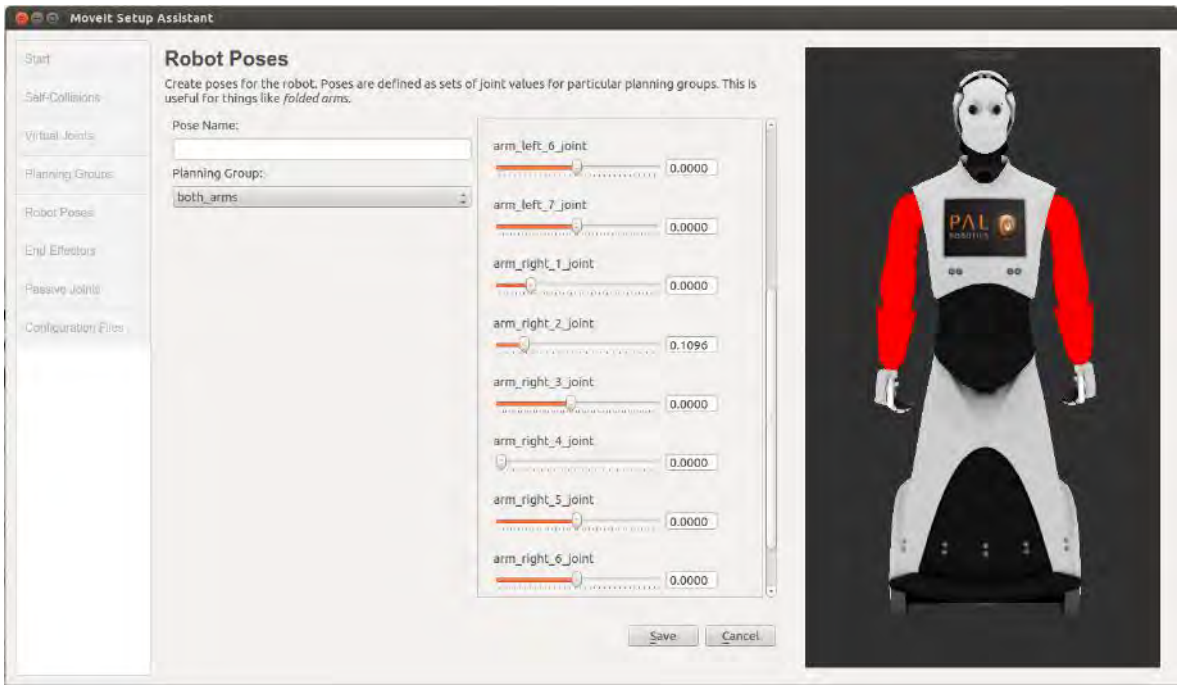

Figura 4.4: Interfaz de MoveIt Setup Assistant.

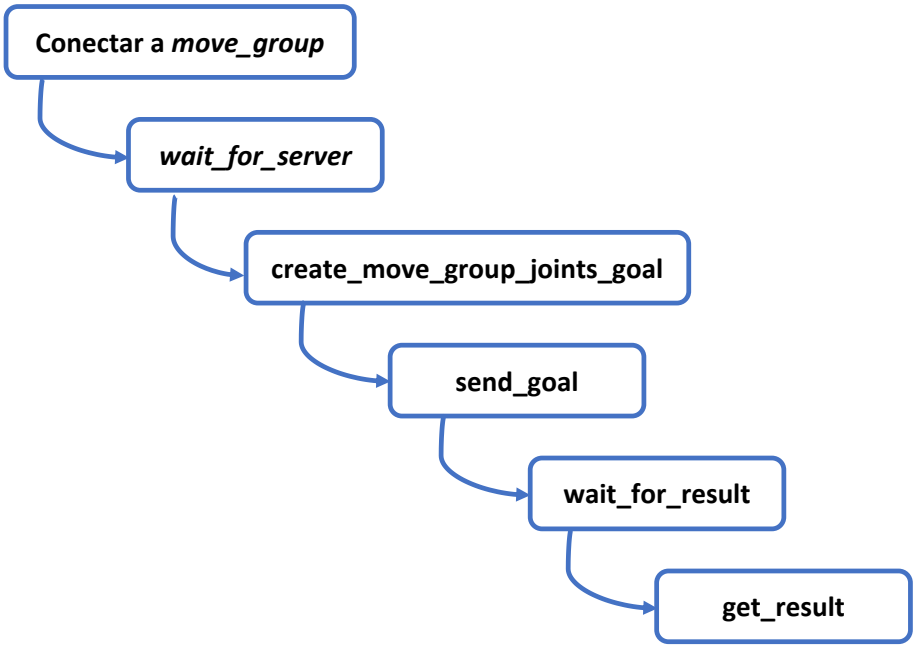

Figura 4.5: Flujo de acciones para la ejecución de una postura.

4.2. Establecimiento de posturas en REEM

Una vez obtenidos los elementos necesarios para que tanto la arquitectura NAOTherapist como el robot REEM funcionen, el principal hito era conocer cómo enviar órdenes al robot a través del código Python. Examinando el código fuente (escrito en Python) se estableció una manera de enviar una posición concreta al robot, a través de una serie de funciones que se deben ejecutar secuencialmente. En la Figura 4.5 se pueden ver los pasos a seguir, donde:

1. **Conexión a move-group():** la primera vez es necesario generar una conexión con ROS mediante la acción *actionlib.SimpleActionClient*, lo que permite conectar con la interfaz del robot ejecutada en el simulador. Esta conexión queda abierta y no es necesario volverla a realizar cada vez que se quiera enviar una instrucción al robot.

2. **wait-for-server():** una vez realizada la conexión, ROS se queda en espera en segundo plano hasta que se recibe alguna orden.

3. **create-move-joints-goal(joint-names, joint-list, group, plan-only):** este método creará la meta que se desea conseguir. En este caso es una postura, así que recibirá los parámetros necesarios para su ejecución. Estos parámetros son:

   - *joint-names:* conjunto de nombres de las articulaciones que se van a mover.
   - *joint-list:* valor en radianes para el ángulo que debe tomar cada motor de cada articulación. Este array debe coincidir en tamaño con *joint-names*, ya que los valores son asociados a dichas articulaciones.
   - *group:* grupo al que pertenecen las articulaciones que se van a mover. Puede ser un brazo, los dos brazos solos, los dos brazos junto con el torso, o todo el cuerpo.
   - *plan-only:* valor booleano que indica si el robot debe planificar el movimiento o moverse directamente a la postura indicada. En caso de planificarlo pasará de la posición actual a la indicada teniendo en cuenta posibles colisiones con su propio cuerpo.

4. **send-goal(goal):** una vez creada la meta, ésta se envía al robot mediante el comando send-goal.

5. **wait-for-result():** cuando la meta ha sido enviada, el componente ReemComp queda a la espera de recibir una respuesta por parte del robot. Se puede incluir como parámetro el tiempo de espera deseado, pasado ese tiempo el sistema continuará con su ejecución independientemente de que se haya alcanzado la meta o no.

6. **get-result():** por último se recibirá el resultado de la realización de la meta, que permitirá conocer al usuario si dicha meta se ha conseguido o ha fallado.

Siguiendo este proceso se ha conseguido la correcta conexión con el robot y la realización de posturas concretas. Este proceso es básico para la parte de retargeting y se ha integrado en el nuevo componente, tal como se describe en la Sección 4.3.

4.3. Implementación de ReemComp

En esta sección se detalla la implementación del componente ReemComp. Realizadas todas las consideraciones previas, se tiene en este punto la arquitectura NAOTherapist integrada, así como las tecnologías necesarias para el funcionamiento del robot REEM. Con este componente se incluirá el robot REEM en la plataforma NAOTherapist, para posteriormente realizar la experimentación sobre el juego Simon. La morfología del robot está explicada en el Capítulo 2 de este documento, incluyendo los diferentes grados de libertad y articulaciones que se pueden manejar.

Como ya se ha mencionado en el punto anterior, el componente contiene la interfaz Ice en la que se definen las funciones que tiene el componente. En ese punto se explicará la codificación de *ReemComp.py*, que contendrá la codificación de dichas funciones. A su vez se tendrá *interface.py*, que estará compuesto por el código asociado a la interfaz gráfica, y que igualmente se comunica con ReemComp.

Este componente se comunica con el componente Ejecutivo de la arquitectura NAOTherapist.

4.3.1. Métodos de control

En primer lugar, es necesario comprobar el estado del robot. Para ello se han codificado dos métodos que comprueban si el robot está conectado y si está o no en modo simulado.

isSimulated():

El método *isSimulated* comprueba si el robot REEM está simulado o la ejecución se va a realizar sobre el robot real. Este método devolverá **True** en caso de que el robot esté simulado, y **False** en caso de que no lo esté. Por ahora, siempre se obtendrá el valor True ya que el robot real para realizar pruebas se tendrá a disposición más adelante.

isConnected():

El método *isConnected* debe comprobar que ROS esté en ejecución, ya que es imprescindible para que Gazebo se ejecute y REEM pueda estar operativo para recibir instrucciones. Para que ROS esté en ejecución se utiliza la instrucción *roscore*, que contiene una colección de nodos y programas que son requisitos previos para el funcionamiento del sistema. *Roscore* inicializa la *master*, nodo base que proporciona el servicio de nombrado y registro para el resto de los nodos de un sistema basado en ROS. Sin esta instrucción, no se podría utilizar ningún elemento de ROS. Por otra parte, *roscore* también inicializa el nodo *rosout*, encargado de todo el registro de eventos, o *logging*, del sistema.

Para comprobar que REEM está arrancado se utiliza el comando *rostopic*, que permite comprobar qué *topics* hay activos. Un *topic* es un bus sobre el los nodos pueden intercambiar mensajes. Si entre ellos se encuentra el mensaje **/rosout**, ROS está correctamente inicializado. El método devolverá **True** en caso de que el robot esté conectado, y **False** en caso de que no lo esté.

4.3.2. Reproducción de audio y Text-to-Speech

Para controlar la reproducción de audio, se distingue entre la reproducción de un fichero o la reproducción de audio a través de un texto. Para ello se han codificado los métodos **playAudioFile** y **say**.

playAudioFile(file, blocking):

El método *playAudioFile* recibe como parámetros el nombre del fichero y si el audio debe ser o no bloqueante. Un audio es bloqueante cuando se debe reproducir completo para que el componente pueda seguir con su ejecución. En cambio, si el audio es no bloqueante el sonido se reproducirá en un segundo plano mientras continúa la ejecución. Para reproducir el fichero se utiliza la librería *Pygame.mixer*[3]. En caso de que el audio sea bloqueante la propia librería meterá un tiempo de espera correspondiente a la duración del fichero. Este método no devuelve nada.

say(speech, blocking):

El método *say* recibe como parámetros el texto que se desea reproducir y si debe o no ser bloqueante. En primer lugar se comprueba si para dicho texto existe un fichero de audio correspondiente, ya que las principales frases que el robot dice durante una terapia o ejecución de Simon están grabadas en audio con el fin de que la voz sea mas agradable al usuario. Si dicho fichero existe, igual que en el método anterior se hará uso de la librería *Pygame.mixer* para su reproducción.

En caso de no encontrar un fichero de audio que corresponda con el texto a reproducir, es necesario utilizar un sistema de *text-to-Speech* que convierta dicho texto en audio. Para la realización de este proyecto se ha usado el paquete *Pyttsx*[4]. Este paquete contiene un completo *text-to-Speech* en diferentes idiomas que permite enviar un texto a su motor y que éste sea capaz de reproducirlo.

Se ha decidido el uso de este paquete en concreto por su facilidad de uso así como por la fácil configuración del idioma de reproducción, lo que permite que la arquitectura sea utilizada en un mayor número de entornos dependiendo del idioma con cambios mínimos en el código. Es posible controlar si el audio es o no bloqueante a través del propio motor del *text-to-Speech*. Este método no devuelve nada.

[3] http://www.pygame.org/docs/ref/mixer.html Accedido por última vez el 07/02/2016
[4] https://pyttsx.readthedocs.org/en/latest/# Accedido por última vez el 08/02/2016

4.3.3. Interfaz

Para poder comunicar al usuario que una postura está correctamente realizada, el robot NAO dispone de una serie de *leds* en los ojos que NAOTherapist aprovecha para que mediante un gradiente de color verde informen de lo bien realizada que está una postura. Cuanto más se asemeje a la postura correcta, más brillante será la tonalidad del color verde. En caso de que la postura no se realice correctamente, no se mostrará este color verde y el robot intentará corregir la postura del usuario. En este caso el robot REEM no dispone de leds en los ojos, por lo que a priori esta funcionalidad no se podía replicar. Sin embargo, aprovechando que el robot REEM tiene una pantalla táctil en el pecho que se usa exclusivamente para contenido multimedia, se decidió desarrollar una interfaz gráfica que contuviese la funcionalidad de estos leds en los ojos.

Adicionalmente, el robot NAO tiene diferentes botones físicos por el cuerpo que pueden realizar diversas funciones dentro de la arquitectura, como pausar la ejecución o saltar la comprobación de una postura. Como REEM carece de botones físicos y la arquitectura NAOTherapist otorga funcionalidad a dichos botones, se ha decidido incluir estos botones como botones de la interfaz para mantener toda la funcionalidad posible. En la Figura 4.6 se muestra el resultado de esta parte del desarrollo.

4.3.3.1. Ojos

La interfaz se ha desarrollado en Python al igual que el resto del sistema. Para su creación se ha utilizado la librería *Tkinter*[5]. Es un paquete estándar en el campo de las interfaces gráficas de usuario, muy ligera y ampliamente utilizada. Se ha escogido esta herramienta por la calidad de la documentación sobre la misma, así como por la facilidad para la creación de la interfaz buscada. También se ha utilizado por su compatibilidad con *Openbox*[6], gestor de ventanas para entornos UNIX especialmente diseñado para consumir muy pocos recursos y utilizado en el ordenador multimedia

[5] https://wiki.python.org/moin/TkInter Accedido por última vez el 12/02/2016
[6] http://openbox.org/wiki/Main_Page Accedido por última vez el 12/02/2016

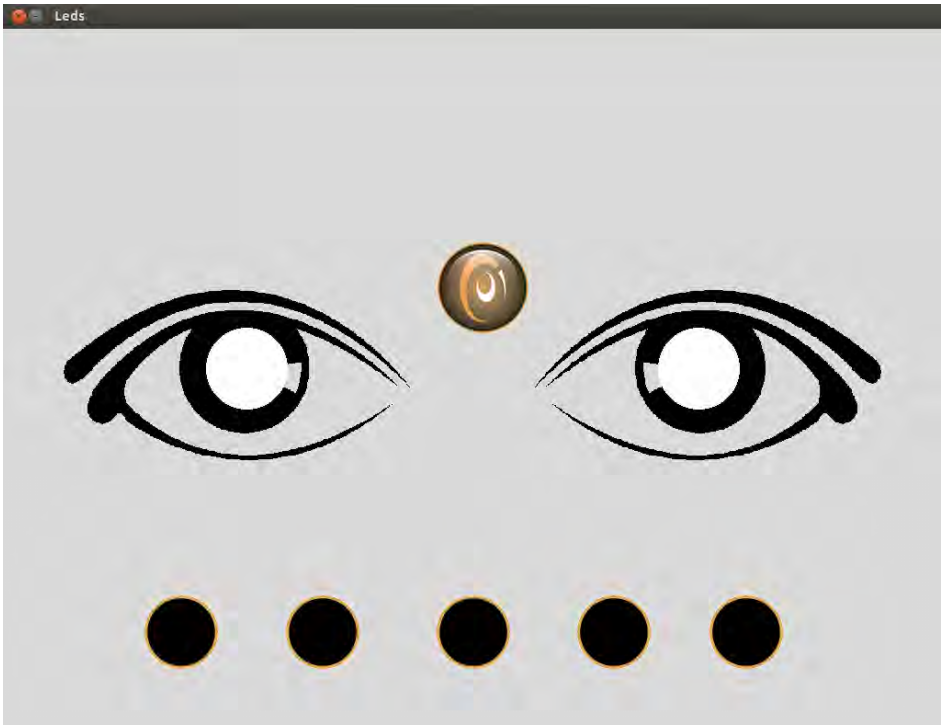

Figura 4.6: Interfaz completa.

que se muestra en la pantalla del pecho del robot. Asegurando esta compatibilidad se sabe que la interfaz podrá ser ejecutada en el robot físico sin ningún problema.

Inicialmente se ha creado una ventana vacía de tamaño 1024x768. Se ha utilizado este tamaño porque coincide con la resolución de la pantalla integrada en el robot, por lo que en el REEM real aparecerá en pantalla completa.

En esta ventana se ha incluido la imagen de unos ojos simples centrados como los que se pueden ver en la Figura 4.6. El hecho de que se hayan elegido unos ojos pese a que aparezcan en la pantalla del pecho del robot es debido a que se ha querido mantener la similitud con el robot NAO. En cualquier caso, como se explicará en la Sección 4.3.3.3, se ha desarrollado una interfaz personalizable en la que se podrían sustituir los ojos por otro elemento manteniendo la misma funcionalidad. Se han utilizado medidas relativas que centren los objetos en pantalla, de forma que una redimensión del tamaño de la ventana afecte lo mínimo posible a la estructura de la interfaz. Se han escogido

específicamente unas imagen que tengan las pupilas en blanco, ya que sobre ellas se van a incluir dos círculos que reflejarán el color que venga de la arquitectura.

Una vez montada la interfaz es necesario comunicarla con el componente del robot, que a su vez es el encargado de comunicarse con el resto de la arquitectura. Para ello se inicializará la ventana vacía en el componente *ReemComp* y se importará la clase de la interfaz para poder acceder a sus métodos, permitiendo el dibujo de los elementos de la misma.

En este punto es importante destacar el hecho de que la interfaz una vez inicializada se queda en una espera activa, es decir, se mantiene en un segundo plano verificando repetidamente una condición como puede ser el cierre la interfaz o que lleguen interrupciones de la misma como pulsaciones de botones o cambios en algún elemento. Esto ha supuesto un problema, ya que tanto la interfaz como el componente del robot se quedan en una espera activa hasta que se reciban instrucciones. Dado que ambas se ejecutan en el componente a la vez, se produce una situación de bloqueo que impide la ejecución de una de las dos partes. Se ha solucionado inicializando la interfaz igualmente en el componente, pero en un hilo de ejecución diferente.

setLeds(ledGroup, intensity)

Una vez integrada la interfaz en el componente, el método *setLeds* del componente es el encargado de recibir del componente Ejecutivo de la arquitectura el color que se debe mostrar. Este método recibe como parámetros el grupo de leds a modificar (*RightFaceLedsRed, RightFaceLedsGreen, RightFaceLedsBlue, LeftFaceLedsRed, LeftFaceLedsGreen, LeftFaceLedsBlue, AllLeds*), y la intensidad de color (valor entre 0 y 1). Todos los grupos de leds se guardan en un array que mantiene el último valor para cada grupo, por lo que modificar un grupo no afecta al valor que tuvieran los demás, a excepción del grupo *AllLeds*, que modifica el valor de todos los grupos.

El valor de la intensidad se recibe como un número normalizado entre 0 y 1. Para poder tomar el color correctamente en la interfaz, es necesario que sea un valor en

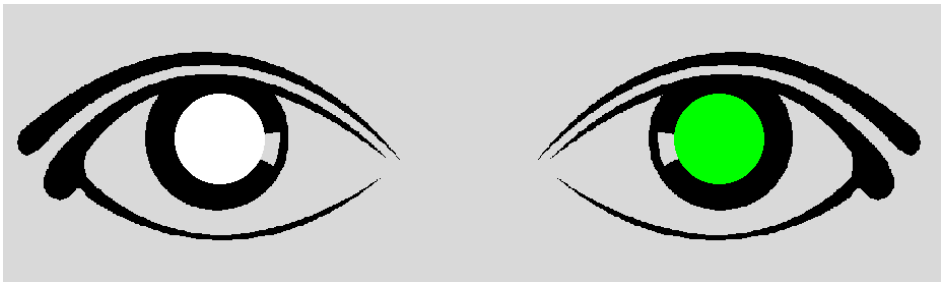

Figura 4.7: Ejemplo de la simulación de los leds de los ojos.

formato *RGB*, por lo que este valor se modifica para que esté comprendido entre 0 y 255. Para verlo más claro se explica el siguiente ejemplo:

Se tiene que *RightFaceLedsRed, RightFaceLedsGreen, RightFaceLedsBlue* tienen una intensidad de 0. Es decir, el color del ojo derecho será negro. Si se llama al método con los siguientes parámetros: *setLeds(RightFaceLedsGreen,1)*, se modificará el componente rojo del color del ojo derecho a 1, que una vez convertido pasará a ser 255. Como los valores de los otros elementos no se pierden, se tiene que el valor del ojo derecho pasa a ser *0,255,0*, con lo que el color del ojo pasaría de ser negro a ser verde.

Una vez establecidos los valores en formato RGB, se llamará al método *updateSetLedsColor* de la interfaz, que recibe como parámetro el array con los valores para cada grupo de leds, modificando el color de los círculos que simulan las pupilas del ojo. El resultado del ejemplo anterior sería el mostrado en la Figura 4.7.

Añadir que los ojos están espejados para facilitar que el usuario asocie bien el brazo que debe colocar.

4.3.3.2. Botones

El robot NAO consta de 6 botones físicos: 3 en la cabeza, 1 en el pecho y 1 en cada pie. En equivalencia, se han añadido 6 botones en la interfaz: uno central que corresponde al botón del pecho y 5 en la parte inferior, que corresponden con el resto de botones mencionados (ver Figura 4.6). El funcionamiento en el componente es muy

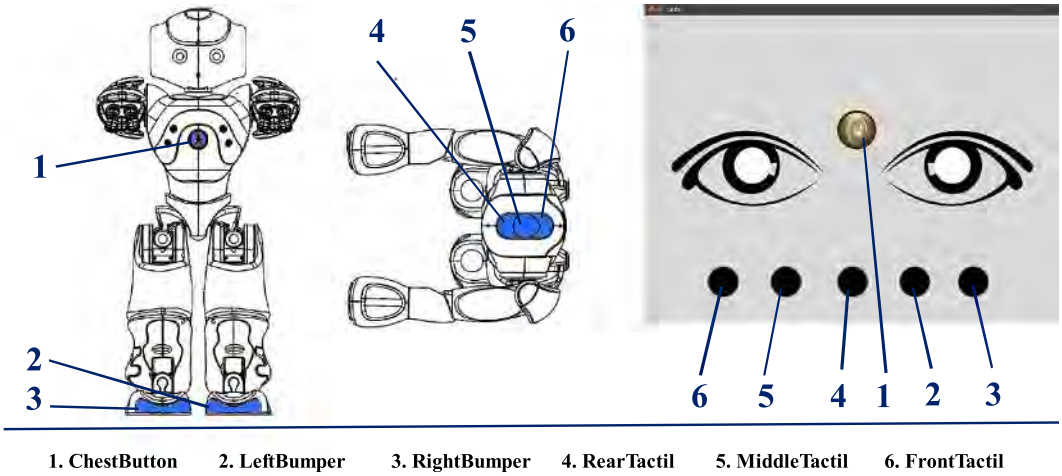

Figura 4.8: Equivalencia entre botones de NAO y REEM.

simple, se utiliza el método *getLastButtonPressed*.

getLastButtonPressed():

Este método guarda el último botón pulsado y lo devuelve. La funcionalidad asociada al botón viene programada según el componente Ejecutivo de la arquitectura por lo que no se ha tenido que desarrollar funcionalidad alguna para los botones. En la interfaz cada vez que se pulsa un botón se actualizará la variable que guarda el último botón pulsado, variable que se devolverá al componente cada vez que éste la solicite. En la Figura 4.8 se puede ver la equivalencia entre los botones físicos del NAO y los implementados en la interfaz para REEM.

4.3.3.3. Skins

Por último, se ha añadido la posibilidad de personalizar la interfaz con *skins* para ayudar a la interacción en función del entorno en el que se ejecute la arquitectura. Por ahora se han añadido las skins *lady* y *doctor* (ver Figura 4.9). Esto da valor a la interfaz a la hora de utilizarlo por ejemplo con niños y diferenciarlo de ambientes más serios.

Figura 4.9: A la izquierda skin lady, a la derecha skin doctor.

4.3.4. Animaciones

Otro punto que favorece la interacción son las animaciones predefinidas. El robot REEM ya traía integrada una animación en la que el robot saluda, pero se ha complementado con algunas animaciones extra que se utilizan en el NAOTherapist original y en el Simon con el NAO. Estas animaciones se ejecutan mediante el uso de la librería *play-motion* que se analizó en la Sección 4.1.4 pero en lugar de utilizar la interfaz gráfica se ha incluido en el código del componente. En la Figura 4.10 se pueden ver los pasos seguidos para la realización de una animación.

El funcionamiento es muy similar al detallado en la Sección 4.2. Se genera una conexión con *play-motion*, librería utilizada para la ejecución de las animaciones, y ROS queda en espera hasta que se reciban instrucciones. Una vez establecida la conexión, se debe generar la meta, que contiene como información el nombre de la animación, y si deberá planificar el movimiento entre posiciones. Si se planifica, el robot intentará evitar cualquier tipo de autocolisión modificando el movimiento si es necesario, mientras que si no se planifica irá directo de una posición a la siguiente. A partir de este punto los pasos a seguir son los mismos que para el movimiento. Se envía la meta a *play-motion*, y se espera la información del resultado de la ejecución.

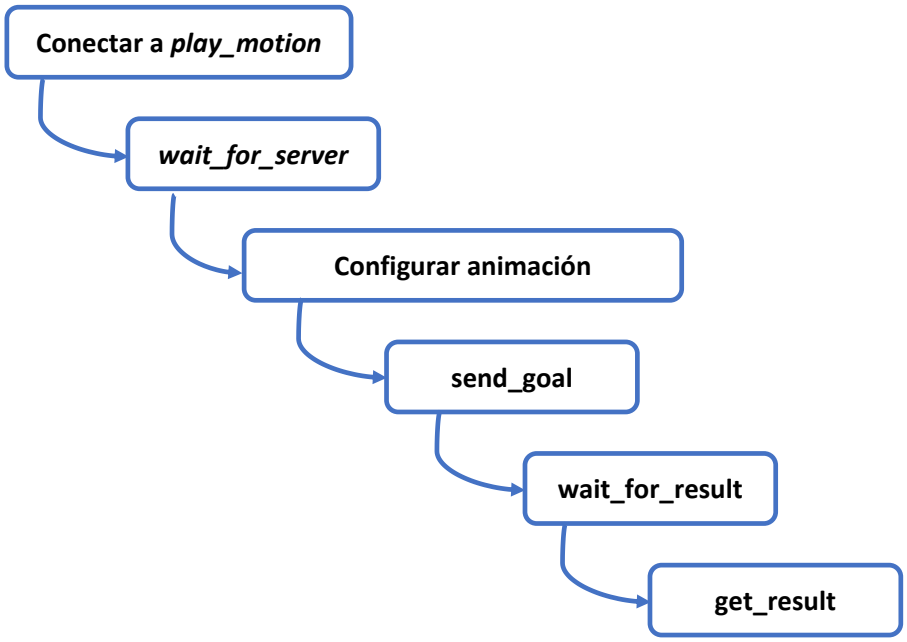

Figura 4.10: Flujo de acciones para la ejecución de una animación.

El funcionamiento de la creación de animaciones se basa en una sucesión de posiciones encadenadas en el tiempo. Por tanto, para la creación de una animación concreta se deben obtener una serie de posiciones, e indicarles el momento en el que deben ser ejecutadas. Una vez indicados los tiempos, al ejecutar la animación el robot irá pasando de una posición a otra en el tiempo indicado, transitando entre posiciones planificando el movimiento de forma autónoma.

execute-animation(name):

Para incluir las animaciones en el componente se utiliza el método *execute-animation*. Este método recibe el nombre de la animación y la ejecuta. Este es un listado de las animaciones que se han tenido que desarrollar para que se adapten a REEM y su morfología concreta:

- **blinking:** simula un pestañeo, los ojos pasan del color actual a negro durante un breve momento y vuelven a su color anterior.

- **close-eyes:** acción de cerrar los ojos, poniéndolos en negro.
- **dance-macarena:** animación que simula el conocido baile de la macarena, junto con la reproducción de la canción.
- **get-stronger:** animación que simula el gesto de estar poniéndose fuerte.
- **hello:** animación que simula un saludo con la mano.
- **inhalation:** acción de respirar. El robot echa la cabeza hacia atrás y abre ligeramente los brazos, a la vez que se reproduce un sonido de inhalación de aire.
- **randomEyes:** utilizada para llamar la atención cuando el usuario se distrae. Realiza un parpadeo rápido en el que los ojos cambian de color entre verde, amarillo, rojo y azul dos veces.
- **rasta:** animación similar a la anterior, pero con un parpadeo más lento y realiza la transición entre colores una única vez.
- **SitSleep:** el robot se sienta y cierra los ojos. En este caso el robot REEM no se puede sentar, por lo que solo realiza la acción de cerrar los ojos (cambiando su color a negro).
- **SitWakeUp:** el robot se levanta, abre los ojos y pestañea 2 veces. En este caso solo se realiza la acción de abrir los ojos (cambiando su color a blanco) y el pestañeo (parpadeo en negro para volver a blanco).
- **Stand:** ejecuta la pose *home*, es decir, cabeza al frente y brazos abajo.

Una muestra de estas animaciones se añade en el Capítulo 5 de este documento.

4.3.5. Retargeting

La última parte del componente es la asociada con el el establecimiento de poses de brazos según lo indicado por esqueletos de Kinect. Para la realización del movimiento se han codificado los métodos *setAnglesFromVision* y *retargeting*.

setAnglesFromVision(angles, mirrored):

El método *setAnglesFromVision* es el encargado de enviar al robot la orden de movimiento. Para ello el método recibirá como parámetros la lista de ángulos que forman la postura deseada y si el movimiento está o no espejado. Si está espejado el robot levanta el mismo brazo que el usuario, si no levanta el brazo contrario puesto que está frente al usuario. Esto para el dominio de Simon es menos relevante, pero en NAOTherapist original si un paciente no consigue poner una postura de forma correcta, el robot actúa de espejo e imita su pose para indicarle como corregirla. La lista de ángulos recibida es la del esqueleto de la Kinect, por lo que no está adaptada a las peculiaridades del robot. Para ello, se debe llamar al método *retargeting* que se explicará a continuación. Una vez obtenidos los ángulos adaptados a REEM se aplican los pasos explicados en la sección 4.2. para enviar la instrucción al robot y que éste ejecute el movimiento.

retargeting(angles, mirrored):

La implementación del método *retargeting* forma una de las partes mas costosas de esta implementación. Como ya se detalló el Capítulo 2 de este documento, el esqueleto de la Kinect obtenido tiene las características ya descritas en el punto 2.4, pero no se pueden utilizar directamente por las peculiaridades del robot, ya que los planos utilizados y las dependencias entre articulaciones son exclusivas de cada robot. Para explicar en detalle este método se va a analizar cada articulación por separado, indicando las transformaciones que se han aplicado y el motivo por el que se ha hecho. Para la realización inicial de las transformaciones este trabajo está basado en el realizado

para el robot NAO, que tiene su base en [Rossignoli 2015].

4.3.5.1. Apertura del hombro

La apertura del hombro no depende de ninguna otra articulación como ocurre con otras, por lo que únicamente se ha utilizado el valor obtenido de la Kinect. Esta articulación toma un valor de 0º cuando está pegado al cuerpo y de 90º cuando está extendido, de la misma manera que en el modelo original mencionado en el capítulo 2. Atendiendo a las consideraciones que se utilizaron para el NAO, se utiliza el valor de la Kinect asociado al seno de plano sagital. El plano sagital es aquel plano que divide el cuerpo en dos partes simétricas de forma perpendicular (ver Figura 2.13, Sección 2.4). Para ambos brazos se ha utilizado el seno de este plano sagital, con la única diferencia de que en el brazo izquierdo se debe invertir el signo respecto al derecho. La única transformación realizada en esta articulación ha sido una reducción de menos de 10º que consiguen tener los brazos ligeramente más abiertos. Esta transformación ha sido necesaria debido a que con los brazos pegados al cuerpo el robot producía una autocolisión y no conseguía tomar la postura.

4.3.5.2. Rotación del hombro

La rotación del hombro funciona de la misma manera que la planteada en el modelo, tomando como 0º cuando el brazo está abajo, 90º cuando el brazo está extendido y 180º cuando el brazo está completamente arriba. Para la rotación del hombro se ha utilizado el plano transversal (ver Figura 2.13, Sección 2.4) que divide el cuerpo por la cintura. En este caso se utiliza el seno del brazo izquierdo y derecho con respecto al plano sin necesidad de un cambio de signo, pero se ha detectado una dependencia con la apertura del hombro.

Cuando el brazo está pegado al cuerpo, el ángulo de apertura vale 0º. En esa situación se ha comprobado que la pose del robot respecto a la pose detectada por la Kinect del usuario difieren en un ángulo de 90º, por lo que el brazo se muestra girado.

Sin embargo, cuando la apertura del brazo es 90º el brazo sí que se muestra correctamente, así que no basta con girar 90º la articulación simplemente. Para solucionar este problema se ha añadido una pequeña fórmula (1) que da mayor o menor peso a la implicación de la apertura del hombro en el giro. Dicha fórmula es la siguiente:

(1)

$$(1 - \frac{d}{\frac{\pi}{2}}) * (\frac{\pi}{2} - 0{,}10)$$

Donde d es el valor de apertura del hombro una vez calculado como se indica en el punto anterior. Esta fórmula se suma al valor obtenido del plano, de forma que si se tiene el brazo pegado al cuerpo ($d = 0$) la fórmula sumará un valor cercano a 90º corrigiendo la postura cuando el brazo esté abajo. En caso de tener el brazo abierto ($d = 1$), la fórmula devolverá un valor cercano a 0º y la apertura no influirá en el valor de la rotación del codo, quedando la postura correcta con el brazo tanto abierto como cerrado. El valor de 0,1 restado en última instancia se ha comprobado que es la media de error para que la pose sea correcta, por lo que al restarlo las poses son más acordes a la realidad.

Finalmente se ha añadido una pequeña corrección en la que el valor obtenido se multiplica por 1,3 para ajustar la postura, ya que el brazo al situarlo en 90º no quedaba recto, si no que queda ligeramente inclinado hacia abajo. Esta pequeña corrección se debe a la morfología propia del robot, ya que el hombro tal y como está diseñado no forma un ángulo de 90º respecto al tronco, si no que tiene una ligera inclinación como se puede ver en la Figura 2.7 de la la Sección 2.3 de este documento.

4.3.5.3. Apertura del codo

La apertura del codo también coincide con la especificada en el modelo seguido, siendo 0º el codo extendido y 90º el codo flexionado. Como sucede con la apertura del hombro, la apertura del codo no depende tampoco de ninguna otra articulación. En este caso, igual que se utilizaba en el NAO, se usa el coseno del plano sobre el

Figura 4.11: Transformación de la apertura del codo. Brazo derecho correcto (con transformación), brazo izquierdo incorrecto (sin ella)

ángulo del codo [Rossignoli 2015] invertido, ya que por las diferencias entre el propio plano y los planos sobre los que se han creado las articulaciones del robot REEM el brazo quedaba en sentido inverso al esperado. Además, ha sido necesario aplicar una transformación en el que se restan 180º al valor del coseno. En la Figura 4.11 se puede observar cómo el brazo izquierdo del robot (a la derecha en la imagen) no tiene aplicada dicha transformación mientras que el brazo derecho si lo tiene, siendo correcta la pose del brazo derecho.

4.3.5.4. Rotación del codo

La rotación del codo es la articulación más problemática del robot, y ha necesitado diversas transformaciones hasta llegar a ser funcional. En el modelo original ya se detectó una dependencia entre la rotación del codo con la posición del hombro. Para ello se desarrolló la fórmula detallada en la Sección 2.4.2 de este documento (1). Gracias a la utilización de esa fórmula en el robot NAO es posible conseguir una correcta

adaptación de la rotación del codo, pero en REEM no es suficiente por la morfología propia del robot y la dependencia entre articulaciones a la hora de realizar la rotación, por lo que se han tenido que aplicar diversas transformaciones para su ajuste. Estas transformaciones se han obtenido de manera empírica siguiendo un proceso de prueba y error. En un primer momento se intentó calcular todas las variantes posibles mediante combinaciones de valores, pero la tasa de poses correctas apenas sobrepasaba el 50 % por lo que se tuvo que descartar un cálculo tán sistemático.

(1)

$$\frac{ac}{\frac{\pi}{2}} + b(1 - \frac{c}{\frac{\pi}{2}})$$

Recordando que:

- a = Ángulo de rotación del codo calculado mediante el vector que une el hombro a la cadera.
- b = Ángulo de rotación del codo calculado mediante el vector que une los hombros.
- c = Ángulo de apertura del hombro.

En primer lugar, a la fórmula original se le ha tenido que añadir la misma transformación que se añadió a la rotación del hombro, ya que se debe tener en cuenta la apertura del hombro también en la rotación del codo. En la primera imagen (1) de la Figura 4.12 se puede observar como el brazo izquierdo del robot (a la derecha en la imagen) no tiene aplicada dicha transformación mientras que el brazo derecho si lo tiene, siendo correcta la pose del brazo derecho. La fórmula utilizada es la siguiente (2):

(2)

$$(1 - \frac{d}{\frac{\pi}{2}})\frac{\pi}{2}$$

Con esta corrección se han solucionado todos los problemas siempre y cuando el brazo esté extendido, es decir, la apertura del codo sea 0º. Cuando la apertura

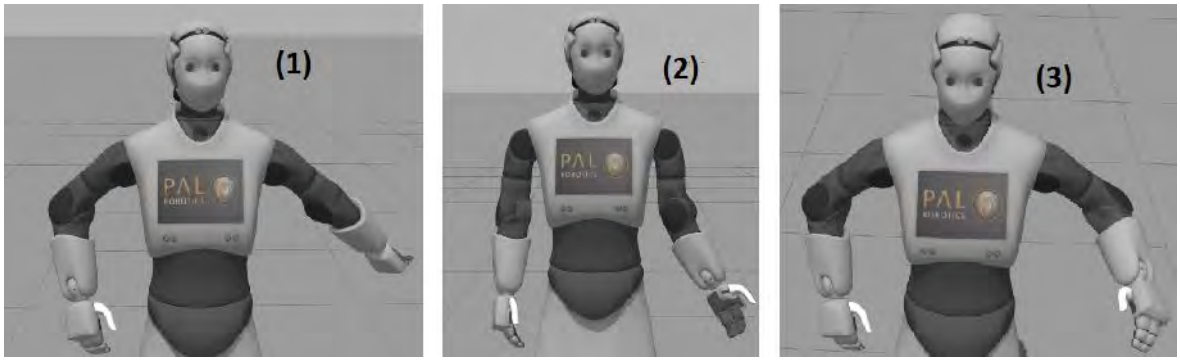

Figura 4.12: (1) Primera transformación (2) Segunda transformación (3) Tercera transformación.

del hombro es 0º la fórmula (1) tomará un valor cercano a 90º y se aplicará dicha corrección en la rotación. Sin embargo, existe un segundo problema cuando el brazo está extendido ($d = 90º$). Si se observa la segunda imagen (2) de la Figura 4.12, cuando tanto la apertura del hombro como la apertura del codo toman un valor de 90º, existe un desfase de 90º (en el brazo derecho se puede ver la posición correcta, mientras que el izquierdo no se le ha aplicado la segunda transformación). Para solucionar este desfase se ha aplicado la siguiente fórmula (3):

(3)

$$\frac{d}{\frac{\pi}{2}} * (\frac{e}{\frac{\pi}{2}} * \frac{\pi}{2}) = \frac{2ed}{\pi}$$

Donde $d = Apertura\ del\ hombro$ y $e = Apertura\ del\ codo$. Con esta transformación, si el brazo está abierto (d = 90º) y el codo está flexionado (e = 90º) se añade un desfase de 90º, que corrige la posición de la postura mostrada en la tercera imagen (3) de la Figura 4.12. Este desfase se ha tenido que restar, debido a que si se sumaba al total se obtenía un desfase de 180º.

Una vez aplicadas todas las transformaciones, las posturas eran correctas pero en el orden inverso, por lo que se ha cambiado de signo a todo el conjunto, obteniendo la siguiente fórmula final:

(4)

$$RotC = \frac{ac}{\frac{\pi}{2}} + b(1 - \frac{c}{\frac{\pi}{2}}) - \frac{\pi}{2} + d + \frac{de}{\frac{\pi}{2}}$$

Se ha utilizado para toda la explicación la fórmula expandida para facilitar la comprensión de las transformaciones realizadas, pero en la versión final del componente se ha simplificado lo máximo posible esta fórmula, que puede verse a continuación (5):

(5)

$$b + d + \frac{4(ac - bc + de) - \pi^2}{2\pi}$$

El resultado final es mucho más complejo de lo esperado en un inicio en comparación con la transformación realizada para el robot NAO, por lo que cabe la posibilidad de que la elección de los planos sobre los que se ha trabajado no haya sido la elección más acertada. Al margen de los planos, la morfología del robot puede hacer que el retargeting sea realmente complicado. Un pequeño cambio como la conexión de la desviación del hombro respecto al torso puede generar, como así ha sido, muchisimos problemas a la hora de realizar el retargeting. Se mencionará en trabajos futuros un posible estudio sobre el uso de otros planos para simplificar el *retargeting* de este robot con la arquitectura NAOTherapist.

4.4. Integración Simon

La adaptación del juego Simon [Turp et al. 2015] se encuentra incluida en la arquitectura NAOTherapist, concretamente en el componente *PELEA* y el *Ejecutivo*. Esta adaptación está escrita en el lenguaje para describir problemas de planificación *PDDL* [Knoblock et al. 1998]. Este dominio está formado por una serie de acciones que en función de diversos parámetros y precondiciones dan como resultado distintas acciones. Para este proyecto no se ha entrado en detalle en la definición del dominio en sí, ya que era un trabajo previo ya incluido en NAOTherapist.

Sin embargo, sí que ha sido necesario conocer su funcionamiento porque se han creado problemas específicos para su ejecución con el robot REEM. En estos problemas se ha definido el número de secuencias que va a tener la ejecución, además del número de poses que se pueden utilizar durante el juego. En la evaluación del sistema del Capítulo 5 se han aplicado estos problemas, con diferente número de poses y secuencias, para probar el correcto funcionamiento del juego en el robot REEM.

Adicionalmente, se ha añadido al problema una pausa que no existía. En el NAOTherapist original se definen pausas cada cierto número de ejercicios para que el paciente descanse, realizando alguna animación suave como respirar o el baile de la macarena. Dado que se han realizado animaciones, se ha decidido añadirlas en la ejecución completa para demostrar la funcionalidad del componente en su totalidad.

Por último, mencionar que de la misma manera que ocurre con los ojos, las poses realizadas se muestran de manera invertida, para que el usuario que esté interactuando con el robot imite la postura como si estuviera delante de un espejo. Esto es importante ya que facilita la comprensión de la postura que debe tomar el usuario.

Capítulo 5

Evaluación del sistema

En este capítulo se tratará la evaluación del sistema, donde se realizará una serie de experimentos que abarquen todas las funciones programadas en este trabajo. Además, se utilizarán estas pruebas para demostrar que los objetivos planteados en el Capítulo 1 de este documento han sido cumplidos.

Todos los vídeos de evaluación de este proyecto se han subido al canal oficial del proyecto NAOTherapist. Las pruebas de las secciones 5.3, 5.4 y 5.5 hacen referencias a vídeos de este canal subidos para tal efecto.

https://www.youtube.com/user/NAOTherapist/videos

5.1. Evaluación del retargeting

En este apartado se muestran los resultados de la evaluación del sistema de *retargeting* implementado. Para realizar las pruebas se ha utilizado una herramienta auxiliar, *RetargetingHelper* (ver Figura 5.1). Esta herramienta ofrece un listado de las poses disponibles en la arquitectura NAOTherapist, mostrando para cada una de ellas los valores correspondientes del esqueleto de la Kinect. Si se envía la postura, se llamará al método del retargeting del robot y éste ejecutará la pose.

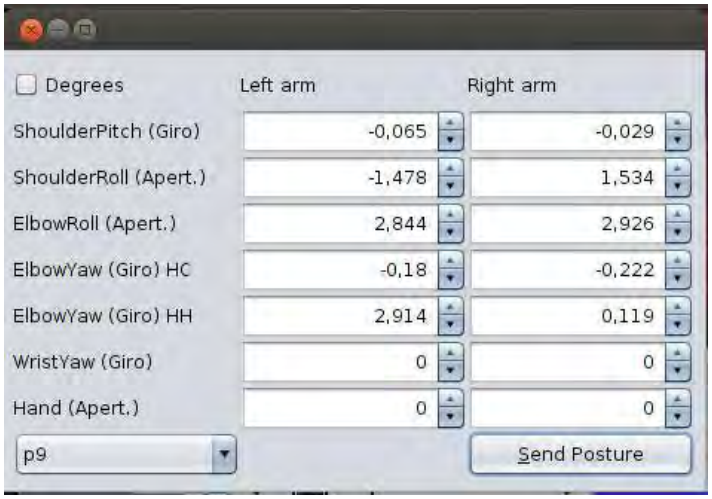

Figura 5.1: Herramienta retargetingHelper.

Se ha modificado el script de ejecución para lanzar a la vez el simulador Choregraphe de NAO y el simulador Gazebo de REEM. Para esta prueba se ha utilizado un conjunto de 7 poses representativas simultáneamente en NAO y REEM utilizando la herramienta retargetingHelper. En esta herramienta se varía la pose, que inmediatamente se refleja en los simuladores de ambos robot.

El objetivo de esta prueba es seleccionar un conjunto de poses y evaluar el método de retargeting en comparación con el del robot NAO. Se trata de un análisis visual, ya que que lo que se pretende es que la postura sea similar, pero debido a la morfología propia de cada robot y las transformaciones de cada retargeting particular hacen imposible medir la calidad de la postura de una forma exacta. En la Figura 5.2 se pueden observar los resultados obtenidos.

De esta colección de poses se han extraído diferentes conclusiones. En primer lugar, debido a la morfología propia de cada robot se observan diferencias difíciles de tratar. REEM tiene los brazos considerablemente más largos en relación con el cuerpo, lo que hace que en imágenes como (2) se tenga la sensación de que los brazos están más abiertos que en el NAO. Debido a la desviación del hombro respecto al tronco ya comentada en anteriores capítulos, se observa que en la imagen (3) el robot REEM

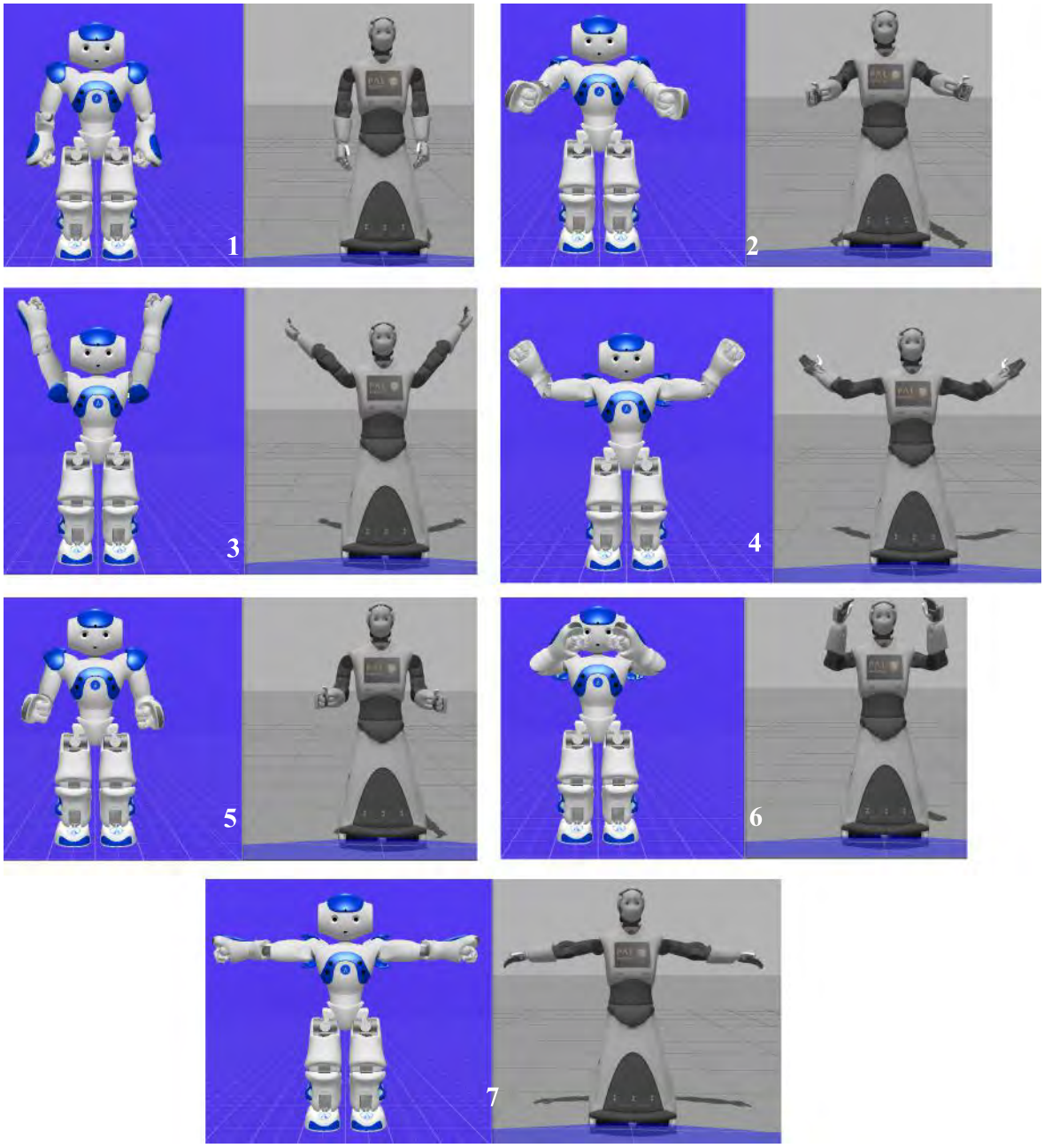

Figura 5.2: Comparativa de poses entre NAO y REEM utilizando retargeting.

no consigue cerrar tanto los brazos como el NAO. Por el mismo motivo la imagen (6) también muestra diferencias, además de que la cabeza de REEM es mucho más pequeña que la del NAO.

Por otra parte, no todos los problemas se deben a la morfología del robot. Pese a las múltiples transformaciones realizadas en el codo, el retargeting aún podría mejorarse, ya que en posturas como la (4) en la que tanto la apertura del hombro como la del codo son cercanas a 90º existe una leve desviación en la postura. Dependiendo de la postura concreta esta desviación se percibe en mayor o menor medida. Los tiempos de respuesta y la velocidad de movimiento son sensiblemente más lentos en REEM que en NAO, pero debido a las propias limitaciones de las herramientas facilitadas por PAL Robotics no es posible mejorar estas características actualmente.

Tras estos resultados se concluye que el retargeting del REEM funciona correctamente, aunque aún hay espacio para mejorar. Con este experimento se demuestra el cumplimiento de los objetivos que implican movimiento del robot y transformación de los ángulos para el robot REEM.

5.2. Evaluación de la interfaz gráfica

En este apartado se incluyen todas las pruebas relacionadas con la interfaz.

5.2.1. Ojos luminosos en pantalla

Como ya se ha explicado en el Capítulo 4 de este documento, el robot utilizará los leds de los ojos para dar *feedback* al usuario que indique si la postura que está haciendo es correcta o debe corregirla. Para ello se utilizará un gradiente de color verde, que será más brillante cuanto más cerca se esté de la posición correcta. Esta pista visual es independiente de cada brazo, por lo que si solo se tiene uno de los dos brazos mal colocado, solo se reflejará en un ojo dicho fallo. Si la pose es completamente incorrecta

el ojo se mostrará en color blanco.

Esta prueba se ha realizado utilizando dos poses diferentes. El sistema comprobará la postura del usuario situado enfrente de la Kinect y la comparará con la pose del robot, indicando en todo momento en la interfaz si ésta es o no correcta. La postura correcta se mostrará en la interfaz del robot, mientras que la postura del usuario se reflejará en el esqueleto que se está obteniendo en ese mismo momento de la Kinect.

Para cada una de las poses se ha realizado una serie de pruebas unitarias con las que se demostrará el funcionamiento de la interfaz.

5.2.1.1. Pose A

La primera pose evaluada corresponde con los brazos en la cintura. En la Figura 5.3 se muestran los resultados de dicha evaluación. Como se puede ver en la imagen (1) la postura es exacta y el sistema la da por correcta. A medida que la pose del usuario se aleja de la pose correcta, se observa que la intensidad del color verde de los ojos va disminuyendo (imagen 4).

Aunque el sistema las empiece a marcar en verde, estas posturas no cuentan como correctas y el robot insistirá en corregir la postura. Esto es debido a que la arquitectura está diseñada para un entorno terapéutico, por lo que es necesario que la postura que se muestre en el usuario coincida en gran medida con la postura deseada. Sin embargo, como Simon se centra también en entretenimiento, se puede aumentar el umbral de permisividad de las poses para este fin.

En la imagen (2) se puede ver una pose completamente incorrecta, por lo que los ojos se muestran en blanco. Por último se demuestra que cada ojo es independiente por si mismo, como se puede ver en la imagen (3).

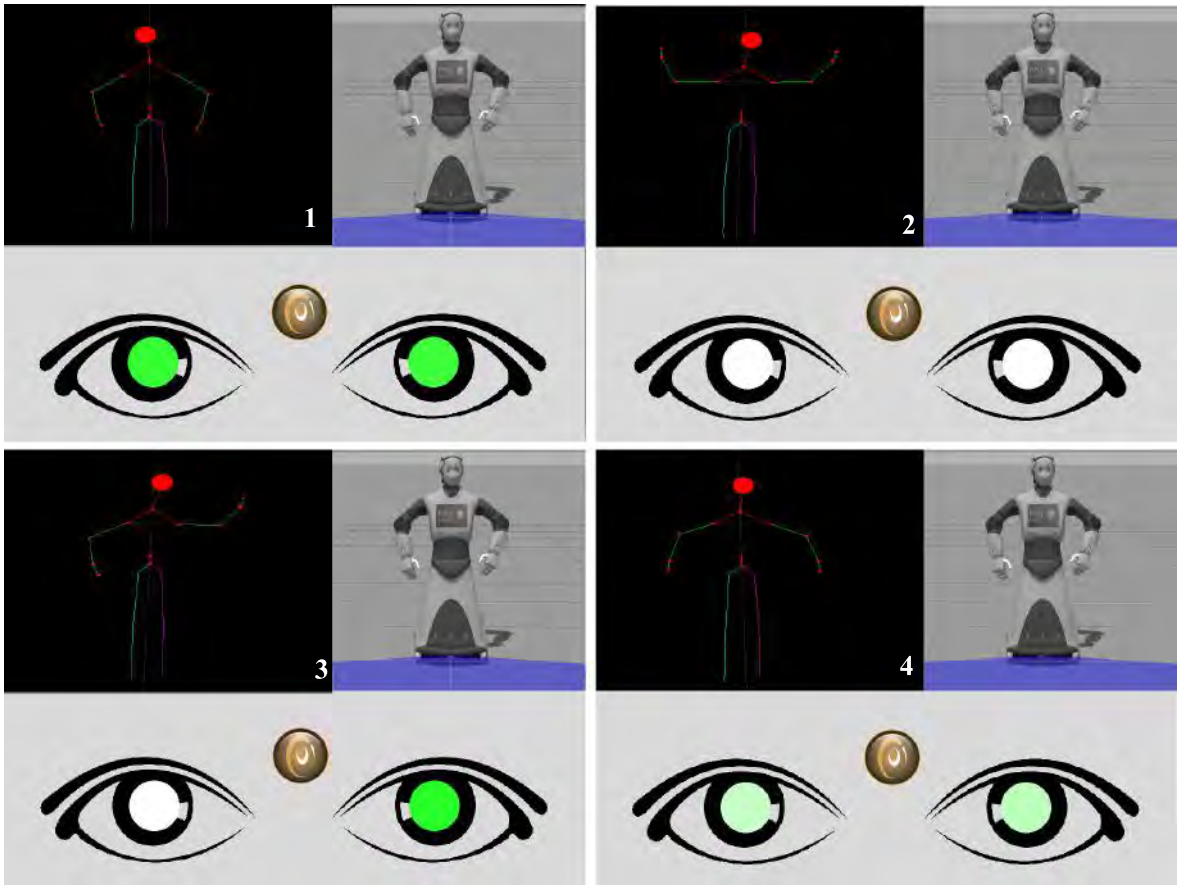

Figura 5.3: Evaluación de la pose A, con esqueleto kinect, el REEM y los ojos, en distintas posturas.

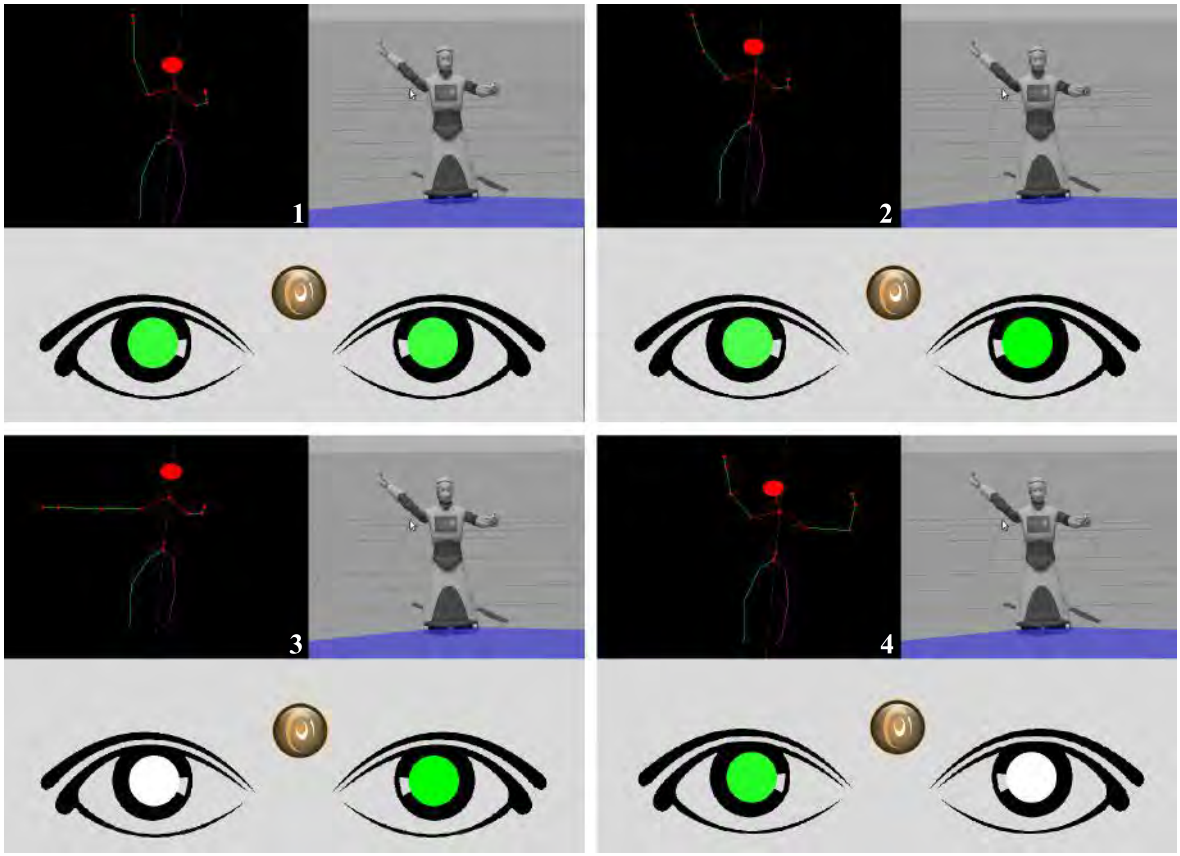

Figura 5.4: Evaluación de la pose B, con esqueleto kinect, el REEM y los ojos, en distintas posturas.

5.2.1.2. Pose B

Para la segunda postura se ha elegido una pose no simétrica (ver Figura 5.4). De la misma manera que ocurre con la pose anterior, se han tomado pruebas unitarias con distintos grados de acierto, en uno o en los dos ojos.

Se ha conseguido el funcionamiento esperado, gracias al cual se puede suplir la falta de leds en los ojos del robot. Destacar que al realizarse directamente sobre la interfaz en lugar de sobre un robot físico, la transición entre colores es más suave que la que hay en la arquitectura original con el NAO.

5.2.2. Botones en pantalla

Para comprobar el funcionamiento de los botones y demostrar su correcta comunicación con el resto de la arquitectura se ha realizado una prueba utilizando el botón *FrontTactil*, que corresponde con el primer botón empezando por la izquierda en la interfaz.

En NAOTherapist, este botón tiene la funcionalidad de pausar la ejecución del sistema cuando se pulsa, y de reanudarla si se vuelve a pulsar. La prueba muestra la traza del componente Ejecutivo (Executive) de la arquitectura, que mostrará la pausa de la sesión cuando se pulsa el botón, y la reanudación del mismo al volver a pulsar.

La traza que se muestra al pulsar el botón en el componente es la siguiente:

```
Last button pressed set to FrontTactil
->getLastButtonPressed: frontTactilTouched
```

Esta acción informa al componente Ejecutivo que se ha pulsado el botón y ejecuta la acción correspondiente, que en este caso es pausar la ejecución. La traza correspondiente al Ejecutivo se puede ver a continuación:

```
1 DETECT-PATIENT [‘nao’, ‘pt00’]
2 IDENTIFY-PATIENT [‘nao’, ‘pt00’]
3 GREET-PATIENT [‘nao’, ‘pt00’]
4 START-TRAINING [‘nao’, ‘loc_training’, ‘pt00’, ‘loc_show’, ‘ses10’]
Emergency: 0
5 PAUSE-SESSION [‘nao’, ‘pt0’, ‘ses1’]
6 RESUME-SESSION [‘nao’, ‘pt0’, ‘ses1’]
7 INTRODUCE-EXERCISE [‘e100r’, ‘nao’, ‘pt00’, ‘ses1’]
8 START-EXEERCISE [‘e100r’, ‘nao’, ‘pt00’, ‘stand_up’]
9 EXECUTE-POSE [‘p_id0’, ‘e100r’, ‘p6’, ‘t1’, ‘nao’, ‘pt00’, ‘stand_up’]
```

Cada uno de los puntos que se muestran en esta traza corresponden con acciones que el componente Executive descompone en acciones de bajo nivel (acciones que han sido programadas en el componente ReemComp). El botón de pausa se presiona durante la acción *START-TRAINING*, lo que causa que se pause la sesión indefinidamente. Al volver a pulsar el botón, se ejecuta la acción *Resume Session* y la sesión continúa sin problemas.

Con estas pruebas se demuestra el cumplimiento del objetivo de implementación de una interfaz gráfica. La alternativa de utilizar la pantalla táctil para suplir la ausencia de luz en los ojos y botones físicos ha funcionado y todo se ha conseguido conectar correctamente sin problemas.

5.3. Animaciones y reproducción de audio

Para la realización de este experimento se ejecutan dos animaciones en el contexto de los descansos dentro de Simon y se observa como se realizan.

El resultado de estas pruebas se puede observar en el vídeo **Animaciones - Evaluación REEM** que se encuentra disponible en el canal de Youtube mencionado al principio de la sección.

En este vídeo se muestra la animación "¿Quieres ponerte fuerte?" y "Macarena" utilizadas en NAOTherapist y hechas originalmente por el NAO. Se han seleccionado estas dos animaciones porque van acompañadas de la reproducción de un fichero de audio. Gracias a esto, queda demostrado el funcionamiento tanto de las animaciones como de la reproducción de audio. Como añadido, se puede observar que la reproducción de audio es no bloqueante, por lo que al empezar la canción de la macarena el robot empieza a moverse, por ejemplo.

La suavidad de las animaciones como norma general es buena, ya que el robot planifica el movimiento de forma correcta. Sin embargo, en animaciones como la macarena que son más rápidas si se nota una mayor brusquedad al transitar entre movimientos.

En conclusión, se han integrado todas las animaciones existentes para el robot NAO y se deja estudiado el procedimiento para futuros añadidos al respecto.

5.4. Text-to-Speech

Para esta prueba se utilizará un texto para el que no exista un fichero de audio correspondiente, por lo que el componente deberá hacer uso del sistema de *Text-to-Speech* para su reproducción.

El resultado de estas pruebas se puede observar en el vídeo **Text-to-Speech - Evaluación REEM** que se encuentra disponible en el canal de Youtube mencionado al principio de la sección.

Con esta evaluación junto a la realizada para las animaciones, queda cumplido el objetivo planteado en el Capítulo 1 sobre estas características.

5.5. Simon

La prueba consiste en una ejecución completa de la arquitectura, en la que se aprecia tanto la parte del robot (pantalla) como la parte del usuario (situado enfrente de la Kinect).

El resultado de estas pruebas se puede observar en el vídeo **Simon - Evaluación REEM** que se encuentra disponible en el canal de Youtube mencionado al principio de la sección.

Para esta prueba se ha utilizado un problema modificado del dominio Simon en el que se ha elegido un conjunto de 3 posturas para el juego. La ejecución completa muestra una partida del juego con un usuario real. Esta partida engloba toda la funcionalidad del proyecto, en la que se evidencia la compatibilidad de la arquitectura NAOTherapist con el componente desarrollado, además de la funcionalidad del dominio de juego. También se incluyen animaciones y sonidos que sirven para tratar la interacción con el usuario en todo momento.

Como conclusión, indicar que esta prueba realmente engloba todas las demás, mostrando que la arquitectura con el nuevo componente funcionan en su totalidad. Por tanto, los objetivos marcados para este proyecto se han cumplido.

Capítulo 6

Planificación y presupuesto

En este apartado se detalla tanto la planificación seguida durante el proyecto actual, así como el presupuesto derivado del mismo, incluyendo tanto costes materiales (hardware y software) como costes personales.

6.1. Planificación

Este proyecto se ha planteado con una duración total de un año, comprendido entre el 1 de Febrero de 2015 y el 1 de Febrero de 2016. En esta sección se especificará tanto la metodología seguida como la planificación en sí del proyecto.

6.1.1. Metodología de planificación

Para la planificación de este proyecto se ha decidido seguir una planificación en cascada (Figura 6.1). Esta metodología especifica que el inicio de cada una de las etapas de la misma está ligado a la terminación de la etapa anterior. Pese a que tiene ciertas limitaciones, como es el hecho de la dificultad de añadir cambios en fases avanzadas del proyecto, se ha decidido que es la mejor opción debido a que es una metodología que especifica claramente cada una de las etapas de desarrollo del proyecto. Para el

proyecto que se está realizando, al no ser un proyecto a gran escala como pueda ser el de una gran empresa, la dificultad de añadir nuevos cambios es menor, por tanto es la opción elegida.

Esta metodología en cascada consta de los siguientes elementos, cada uno de los cuales se deberá realizar antes de continuar al siguiente:

- **Análisis de requisitos:** en esta primera fase se analizan los objetivos que se deben cubrir y las necesidades que surgen de los mismos. En este punto se detalla la especificación de requisitos, que contiene la especificación completa del funcionamiento del sistema. Es importante que este punto se realice con el mayor nivel de detalle posible, pues como ya se ha comentado agregar nuevos requisitos en siguientes fases del desarrollo implica volver a realizar todo el proceso, con el consiguiente gasto que conlleva.
- **Diseño del sistema:** una vez detallados todos los requisitos, se divide el sistema en módulos o subsistemas claramente diferenciados. En cada uno de estos módulos debe quedar claro su diseño y que debe hacer en función de los requisitos previos. De esta fase se obtiene un diseño completo de cada uno de los módulos del sistema, con el objetivo de su posterior implementación.
- **Implementación del sistema:** implementación propiamente dicha del sistema en base al punto anterior.
- **Pruebas del sistema:** en esta fase, con el sistema ya implementado, se realizan las pruebas pertinentes para comprobar su correcto funcionamiento, acorde siempre a las especificaciones detalladas en el análisis realizado en el primer punto.
- **Verificación y mantenimiento del sistema:** pruebas realizadas por el usuario final, que debe dar el visto bueno y verificar que el sistema funciona como se había acordado. En este punto se daría por finalizado el proyecto, aunque en nuestro caso concreto se realiza también un último paso de documentación de todo el proyecto.

Un diagrama de la estructura puede verse en la figura 6.1.

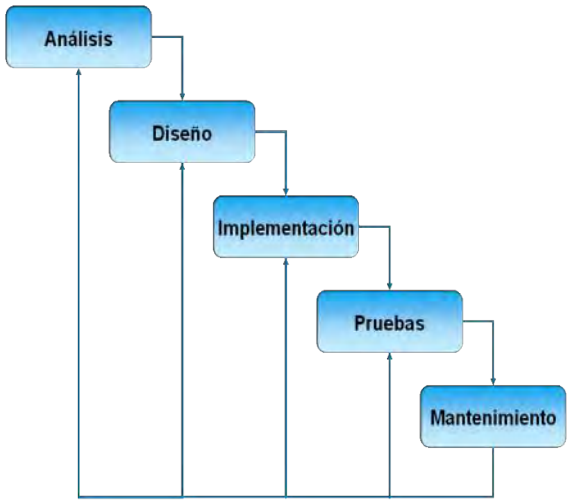

Figura 6.1: Metodología de desarrollo en cascada

6.1.2. Planificación

Como se ha mencionado al inicio de este apartado, el proyecto tiene una duración de un año, comprendido entre el 1 de Febrero de 2015 y el 1 de Febrero de 2016. Durante el transcurso del año se deben completar todas las fases definidas por la metodología en cascada, explicadas en el punto anterior. La especificación resultante de la planificación que se debe seguir en el desarrollo del proyecto se puede ver en la Tabla 6.1.

En el diagrama de Gantt que se muestra en la Figura 6.2 se puede observar de manera gráfica la planificación del proyecto. Se ha planificado una importante cantidad de tiempo para la fase de análisis inicial por la dificultad de hacer compatibles todas las tecnologías utilizadas, siendo indispensable que no surja ninguna incompatibilidad para poder llevar a cabo el proyecto.

Tarea	Duración (días)	Fecha Inicio	Fecha Fin
Análisis del sistema	**84**	**01/02/2015**	**25/04/2015**
Definición con el tutor de los objetivos del proyecto	7	01/02/2015	07/02/2015
Análisis e investigación NAOTherapist/RoboComp	18	08/02/2015	25/02/2015
Análisis e investigación ROS	19	26/02/2015	16/03/2015
Análisis e investigación Gazebo	10	17/03/2015	26/03/2015
Análisis e investigación REEM	22	27/03/2015	17/04/2015
Especificación de requisitos	5	18/04/2015	22/04/2015
Diseño de casos de uso	3	23/04/2015	25/04/2015
Diseño del sistema	**29**	**26/04/2015**	**24/05/2015**
Definición de la arquitectura del sistema	13	26/04/2015	08/05/2015
Componente ReemComp	7	09/05/2015	15/05/2015
Interfaz de usuario	9	16/05/2015	24/05/2015
Implementación del sistema	**133**	**25/05/2015**	**24/10/2015**
Integración de ROS en NaoTherapist	18	25/05/2015	11/06/2015
Comandos simples con Python	17	12/06/2015	28/06/2015
Implementación retargeting	50	29/06/2015	17/08/2015
Reproducción de audio	12	18/08/2015	29/08/2015
Desarrollo interfaz	25	30/08/2015	23/09/2015
Comunicación de ReemComp con la interfaz	18	24/09/2015	11/10/2015
Text to speech	13	12/10/2015	24/10/2015
Pruebas del sistema	**29**	**25/10/2015**	**22/11/2015**
Definición de entorno de pruebas Simon	16	25/10/2015	09/11/2015
Ejecución de pruebas	13	10/11/2015	22/11/2015
Verificación del sistema	**12**	**23/11/2015**	**04/12/2015**
Comprobación de escenarios de prueba y ejecución por parte del tutor	8	23/11/2015	30/11/2015
Revisión y limpieza final del código	4	01/12/2015	04/12/2015
Documentación	**365**	**01/02/2015**	**01/02/2016**
Revisión de la memoria	7	23/01/2016	29/01/2016
Videos de demostración	2	30/01/2016	31/01/2016

Tabla 6.1: Planificación del proyecto

Figura 6.2: Diagrama de Gantt con la planificación del proyecto.

MES	HORAS/SEMANA	TOTAL
Febrero 2015	5	20
Marzo 2015	5	20
Abril 2015	10	40
Mayo 2015	10	40
Junio 2015	20	80
Julio 2015	20	80
Agosto 2015	3	12
Septiembre 2015	15	60
Octubre 2015	10	40
Noviembre 2015	10	40
Diciembre 2015	5	20
Enero 2016	10	40
TOTAL		**492**

Tabla 6.2: Horas semanales trabajadas.

6.2. Presupuesto

En este apartado se detallará el presupuesto asociado al proyecto, evaluando los costes tanto materiales como de personal. Para evaluar el coste de personal se ha seguido la relación de horas semanales trabajadas reflejada en la Tabla 6.2.

6.2.1. Costes de personal

Para calcular los costes de personal se han estipulado los siguientes roles durante el desarrollo del proyecto:

- **Jefe de proyecto:** persona encargada de la dirección del proyecto. Se encarga de hablar con el cliente y de coordinar el desarrollo del proyecto.
- **Analista/Diseñador:** persona encargada de todo el diseño del sistema, así como de la especificación de requisitos y casos de uso necesarios para llevar a cabo el proyecto.
- **Programador:** persona encargada de la implementación del diseño previamente realizado.
- **Tester:** persona encargada del diseño y ejecución de las pruebas realizadas en el

ROL	MESES	SALARIO MENSUAL (€)	COSTE TOTAL (€)
Jefe de Proyecto	12	1.657,85	19.894,2
Analista	3	1.490,03	4.470,09
Programador	7	1.322,17	9.255,19
Tester	1	821,47	821,47
TOTAL			**34.436,95**

Tabla 6.3: Costes de personal.

sistema para comprobar su correcto funcionamiento.

Según los diferentes roles identificados, y teniendo en cuenta los salarios establecidos en el BOE, nº 80 del miercoles 2 de Julio de 2014, sección 3, página 28273, se han calculado los costes de personal que pueden verse en la Figura 6.3.

6.2.2. Costes materiales

En este punto se detalla el coste material asociado al proyecto. Estos costes han sido divididos en costes de hardware y costes de software, y en todos ellos el I.V.A. (Impuesto sobre el Valor Añadido) está incluido. La amortización de los productos se calcula mediante la fórmula 6.1.

$$Amortización = \frac{Precio}{Depreciación} * Uso \tag{6.1}$$

Donde:

- **Precio:** valor en euros del producto.
- **Uso:** valor expresado en meses en los que se ha utilizado el producto.
- **Depreciación:** valor estimado en meses en los que el producto mantiene su valor.

6.2.2.1. Costes de hardware

En la Tabla 6.4 se muestran los costes asociados al Hardware. Todos los elementos que no aparecen en esta tabla pero sí en el entorno operacional definido en el Capítulo 3 han sido facilitados por el laboratorio de *PLG (Planning and Learning Group)* de la Universidad Carlos III de Madrid y por tanto no tienen coste alguno.

ARTÍCULO	PRECIO (€)	USO (MESES)	DEPRECIACIÓN (MESES)	AMORTIZACIÓN (€)
Ordenador portátil Mountain F-13	1.200,00	12	48	300,00
Monitor Dell UltraSharp U2515H	329,00	12	48	82,25
Kinect	144,95	12	48	36,25
TOTAL				**418,50**

Tabla 6.4: Costes de Hardware.

6.2.2.2. Costes de software

En la Tabla 6.5 se muestran los costes asociados al Software.

ARTÍCULO	PRECIO (€)	USO (MESES)	DEPRECIACIÓN (MESES)	AMORTIZACIÓN (€)
Gazebo	00,00	12	60	00,00
Gantt Project	00,00	0,2	60	00,00
ShareLatex	00,00	12	60	00,00
Adobe Acrobat XI Pro	676,39	12	60	169,10
Microsoft Office	149,00	12	60	37,25
TOTAL				**206,35**

Tabla 6.5: Costes de Software.

6.2.3. Costes totales

El conjunto de costes de personal y costes materiales se denomina costes directos. A este coste directo hay que sumarle un 10 % relativo a gastos indirectos de luz, agua, internet, teléfono...

Además, a ese coste se le añadirá un 15 % que contemplará los beneficios que se esperan obtener por la realización del proyecto. El desglose total del presupuesto puede verse en la tabla 6.6.

CONCEPTO	PRECIO (€)
Personal	**34.436,95**
Material	**624,85**
Hardware	418,50
Software	206,35
TOTAL (Coste directo)	**35.061,80**
TOTAL (Coste indirecto)	**38.567,98**
TOTAL (Beneficios)	**44.353,20**

Tabla 6.6: Coste total del proyecto.

El coste total del proyecto asciende a CUARENTA Y CUATRO MIL TRESCIENTOS CINCUENTA Y TRES EUROS CON VEINTE CÉNTIMOS (44.353,20€).

Capítulo 7

Conclusiones y trabajos futuros

En este capítulo se detallan las conclusiones sacadas tras la realización del proyecto, así como las posibles líneas de trabajo futuras que puedan complementar y mejorar el sistema desarrollado en este trabajo.

7.1. Discusión

En esta sección se presentan las principales conclusiones obtenidas a lo largo del documento.

En primer lugar, se ha conseguido establecer una completa compatibilidad del componente desarrollado en la arquitectura NAOTherapist. Esto implica que con la arquitectura actual se pueda utilizar el robot REEM tanto para el desarrollo de terapias como para su aplicación en el dominio Simon. En ambos casos se puede ejecutar completamente una sesión, por lo que su aplicación en un entorno real con pacientes es perfectamente posible, por ejemplo.

Por otra parte, se han establecido diferentes formas de interacción con el usuario. Era una de las principales metas y se ha conseguido tanto ofrecer una interacción visual con la interfaz como una interacción muy directa mediante audio. La conclusión

que se obtiene de este apartado es que si bien funciona bien, en un entorno real se hace necesario utilizar un *text-to-Speech* lo menos robótico posible, lo que mejoraría la interacción con el usuario.

La implementación del retargeting ha resultado mucho más compleja de lo esperado debido a la desviación del hombro respecto al tronco que tiene el robot REEM. Debido a esta situación, la obtención de una transformación correcta que consiga que todas las poses se realicen correctamente ha sido mucho más costosa respecto a las transformaciones originales del NAO. Esta implementación se podría mejorar utilizando un sistema de planos diferentes, pero la dependencia entre articulaciones es mucho mayor que en otros robots como el NAO, lo que hace que independientemente del sistema de planos seleccionado, la implementación no sea sencilla en ningún caso.

En conclusión, se esperaba conseguir con este proyecto un enfoque que dotara al robot REEM de un fuerte componente de interacción humano-robot. En ese aspecto, se ha conseguido una notable implementación que hace que el robot se pueda utilizar con esta arquitectura para múltiples eventos, aparte de las terapias y Simon ya comentados, tales como eventos de promoción o demostraciones.

7.2. Conclusiones técnicas

Tras la realización de este trabajo se pueden obtener una serie de conclusiones, destacando que se han podido cumplir los objetivos planteados al inicio del documento:

- Tanto el *framework RoboComp* como *ROS* se han podido compatibilizar en torno a la arquitectura NAOTherapist, consiguiendo un funcionamiento completo de toda la arquitectura.
- Se han conseguido enviar instrucciones al robot a través del nuevo componente creado en función de los datos recibidos por la arquitectura.
- El *retargeting* se ha podido desarrollar con bastante similitud al movimiento

original, pero cabe destacar que las transformaciones necesarias para su implementación han sido mucho más complejas de lo esperado en un primer momento.

- La interfaz gráfica se comunica con el componente creado y el resto de la arquitectura, por lo que se ha conseguido el objetivo principal que implicaba simular botones físicos y leds de los ojos ya que el robot real no cuenta con ellos y gracias a la interfaz mantienen esta funcionalidad de la arquitectura.

- La reproducción de audio y *text-to-Speech* funciona correctamente, si bien el *text-to-Speech* pese a tener un funcionamiento aceptable entre las opciones encontradas no suena ni mucho menos tan natural como una voz humana, lo que resta cierto valor a la interacción. Las librería de *text-to-Speech* gratuitas tienen por norma general una voz muy robótica, por lo que sería mucho mejor utilizar una librería de pago.

- La ejecución del dominio adaptado de Simon es correcta, además de ser bastante personalizable debido al número de poses que existen en la arquitectura. Esto permite realizar múltiples ejecuciones diferentes con distintos grados de dificultad.

7.3. Trabajos futuros

Tras la realización de este trabajo, una vez comprobadas las posibilidades que ofrece y las capacidades con las que cuenta tanto el robot como la arquitectura, se pueden plantear varias lineas de investigación futuras que mejoren o amplíen este proyecto.

Por un lado, se ha conseguido realizar un *retargeting* bastante logrado respecto a la postura concreta del robot respecto a la postura original del usuario. Sin embargo, hay dos factores que en el robot NAO sí se pueden manejar y en REEM no: la velocidad de movimiento, y que un movimiento sea o no bloqueante. Se ha contactado con la empresa PAL Robotics para comprobar si existe algún método para controlar estos dos parámetros, pero la forma de controlar el movimiento actual no lo permite. En una

Figura 7.1: Robot REEM-C.

futura línea de desarrollo se podrían añadir estos dos campos, que darían al robot una apariencia más dinámica que favorecería la interacción con el usuario.

Por otra parte, el hecho de tener una interfaz gráfica ofrece muchas posibilidades a la hora de añadir contenido o personalización. Incluir un menú que permita seleccionar si se quiere realizar una terapia de rehabilitación o una simulación de Simon, además de cualquier otro dominio que se plantee para la arquitectura, haría de ésta una arquitectura más completa y accesible para el propio usuario.

Otro añadido interesante es la inclusión de este desarrollo para el robot REEM-C[1] (ver Figura 7.1). Este robot es también un producto de la empresa PAL Robotics, con el añadido de que este nuevo modelo tiene piernas. Al utilizar la misma arquitectura y compartir la mayoría de rasgos con el robot REEM utilizado en este proyecto, la adaptación sería bastante directa y así se añadiría otro robot compatible a la arquitectura.

[1] http://pal-robotics.com/es/products/reem-c/ Accedido por última vez 15/02/2016

Apéndice A

Manual de instalación

A.1. Instalación de los componentes

En esta sección se detallan esquemáticamente los pasos seguidos para la instalación de los componentes necesarios para el funcionamiento del nuevo componente ReemComp en la arquitectura NAOTherapist. Indicar que este manual de instalación se ha desarrollado teniendo en cuenta un sistema operativo Ubuntu 12.04, que hasta la fecha es el más moderno compatible con REEM.

A.1.1. Instalación del framework robótico ROS

Dados los requisitos de software del robot REEM, se detalla la instalación de la versión de Ros Hydro.

1. Configurar los repositorios de Ubuntu activando los componentes *Universe / Multiverse*. Se pueden activar en:

```
Menu principal: Sistema > Administrador > Fuentes de Software.
```

2. Configurar `sources.list`

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
```

```
precise main" > /etc/apt/sources.list.d/ros-latest.list'
```

3. Configuración de las claves necesarias para conectar con el módulo principal de ROS:

```
wget \url{https://raw.githubusercontent.com/ros/rosdistro/master/ros.key}
-O - | sudo apt-key add -
```

4. Instalación:

   a) Actualizar índice de paquetes general del sistema:

   ```
   sudo apt-get update
   ```

   b) Instalación completa de ROS Hydro:

   ```
   sudo apt-get install ros-hydro-desktop-full
   ```

5. Inicializar `rosdep` para poder ejecutar algunos componentes importantes de ROS:

```
sudo rosdep init
rosdep update
```

6. Configuración variables de entorno:

```
echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

7. Instalación herramienta `rosinstall`. Herramienta de línea de comandos para descargar código fuente de ROS:

```
sudo apt-get install python-rosinstall
```

A.1.2. Integración del modelo del REEM para Gazebo

A continuación detallamos los pasos para la instalación del simulador Gazebo.

1. Crear workspace, directorio principal que contiene todas las utilidades y herramientas de REEM:

```
mkdir -p ~/reem-sim_ws/src \&\& cd ~/reem-sim_ws/src
catkin_init_workspace
```

2. Descargar archivos de instalación del modelo REEM para Gazebo:

```
wstool init .
wstool merge https://raw.githubusercontent.com/pal-robotics/pal-ros-pkg/master/
reem-sim-hydro.rosinstall

wstol merge \url{https://raw.github.com/ris-controls/ros_control/}
hydro-devel/ros_control.rosinstall
```

3. Actualización de los paquetes REEM descargados:

```
wstool update -j8
```

4. Instalación de los paquetes de ROS necesarios para el funcionamiento de REEM:

```
cd ..
rosdep install --from-paths src --ignore-src --rosdistro hydro -y
sudo apt-get install ros-hydro-ros-control ros-hydro-ros-controllers
```

5. Compilación de todos los paquetes descargados, contenidos en el workspace:

```
catkin_make
```

A.1.3. Configuración variables de entorno

Comprobar que se ha añadido lo siguiente en el fichero $\tilde{/}$.bashrc. Si no es así, añadir información restante.

```
source /opt/ros/hydro/setup.bash
source ~/reem-sim_ws/devel/setup.bash
```

A.1.4. Comprobar funcionamiento

Para probar el correcto funcionamiento del simulador ejecutar en una terminal la siguiente sentencia y se abrirá el simulador Gazebo con el robot REEM.

```
roslaunch reem_gazebo reem_empty_world.launch
```

A.2. Instalación NAOTherapist

NAOTherapist es compatible con Ubuntu 12.04 y Ubuntu 12.04, pero en este manual se expone el método de instalación en la versión 12.04 por la restricción de versión de REEM.

1. Instalar dependencias de NAOTherapist (javabridge y python-weka-wrapper necesarios para uso de weka en python):

```
sudo apt-get install git default-jdk yakuake zeroc-ice34 python-dev python-pip
sudo apt-get install python-pygame
sudo pip install psutil pyglet pyttsx javabridge python-weka-wrapper
```

2. Descargar e instalar choregraphe-suite-1.14.5 y choregraphe-suite-2.1.0[1]. Son necesarias ambas versiones dado que son las utilizadas por los robots NAO del laboratorio del PLG donde se ha desarrollado el proyecto.

   a) Copiar el script Choregraphe dentro de la carpeta raíz de choregraphe dentro de /bin

   b) Editar el nuevo script para quitarle /bin de la última línea (y poner el número de licencia como argumento, si se tiene).

   c) Dentro de /lib borrar todos los enlaces simbólicos de libQt*, libsso* y ibpng12.so.0 para evitar incompatibilidades.

3. Descargar NAOTherapist completo:

```
git clone \url{https://USUARIOBITBUCKET@bitbucket.org/josgonza/naotherapist.git
~/NaoTherapist}
```

4. Crear usuario `NaoTherapist` de esta manera. Sirve para que funcionen los scripts de ejecución de la arquitectura:

```
sudo ln -s ~/ /home/NaoTherapist
```

[1] https://community.aldebaran-robotics.com/

5. Dentro de la raíz de NAOTherapist, ejecutar `transcodeIces.sh`. Este script compila las interfaces *.ice* y las distribuye entre todos los componentes de la arquitectura.

6. Editar versión ice en fichero `config.icestorm.icebox` del componente NTVisionComp

7. En algunos sistemas es necesario recompilar el planificador automático MetricFF dentro de NaoTherapist/Components/PLG/pelea2level/planners/MetricFF con el comando *make*.

8. Añadir estas variables de entorno al final de /.bashrc

```
export AL_DIR=~/Programas/choregraphe-suite-1.14.5-linux64
#export AL_DIR=~/Programas/choregraphe-suite-2.1.0.19-linux64
export LD_LIBRARY_PATH=\$AL_DIR/lib:\$LD_LIBRARY_PATH
export PYTHONPATH=\$AL_DIR/lib:\$PYTHONPATH
export PATH=\$PYTHONPATH:\$AL_DIR/bin:\$PATH
```

9. Cerrar la sesión de Ubuntu y volverla a abrir.

A.2.1. Comprobar funcionamiento

Para comprobar el correcto funcionamiento de la arquitectura se ejecuta el script *light.sh*. Este script tiene distintas opciones:

- -r: la opción -r indica que se ejecutará la arquitectura con el robot REEM. Si no se incluye esta opción, se ejecutará con NAO.

- -o: esta opción sirve para especificar el dominio que se va a ejecutar. En este proyecto sería el dominio de Simon.

- -f: con -f se determina el problema a ejecutar. En este caso, los diferentes problemas que se han creado para Simon.

- -s: este parámetro indica que se está ejecutando en modo simulado. No iniciará el componente de Vision.

Algunos ejemplos de ejecución:

```
NAO: ./light.sh -o dominio.pddl -f problema.pddl
NAO simulado: ./light.sh -o dominio.pddl -f problema.pddl -s
REEM: ./light.sh -r -o dominio.pddl -f problema.pddl
REEM simulado: ./light.sh -r -o dominio.pddl -f problema.pddl -s
```

Apéndice B

English overview

B.1. Introduction

This section describes the context, structure and main goals of the project, which develops a new component for the NAOTherapist project [González et al. 2015] using REEM robot[1]. This first section shows the specific objectives of the project. This component replaces the NAO robot[2] by the REEM robot, keeping compatibility with all the architecture without losing functionality.

Although the NAOTherapist project is focused on rehabilitation therapies, the approach of this work comes from a different point of view, using NAOTherapist architecture to promote human-robot interaction. Enhance this interactive component of the robot with humans allows the use of the architecture in other fields, apart from to rehabilitation therapies. For this project, an adaptation of the popular memory game Simon will be used. This game consists in remembering a sequence of colors and repeat it. For every success, that sequence will become longer until the player fails.

The development of this project requires the integration and installation of two different robotic frameworks into an existing architecture, and communication with

[1] http://pal-robotics.com/es/products/reem/ Last access 11/01/2016

[2] http://www.aldebaran.com/en/humanoid-robot/nao-robot/ Accedido por última vez el 11/01/2016

Task	Carlos Manzano	Andrea Haro
System analysis	50%	50%
System design	50%	50%
System implementation		
Integration of ROS in NAOTherapist	50%	50%
Simple commands for movement in Python	50%	50%
Retargeting	40%	60%
Interface development	70%	30%
Communication ReemComp-interface	80%	20%
Audio play	100%	0%
Text-to-Speech	100%	0%
System evaluation		
Therapy domain	0%	100%
Simon domain	100%	0%

Tabla B.1: Work distribution.

the company which designed the REEM robot, PAL Robotics, to provide development tools. Due to the size of the development process of this project, which also requires several laboratory tests, the work has been made by Andrea Haro and the author of this project. Both students apply their work to different domains, but the first part is inevitably common because there are a high number of dependencies between two developments, so there is at least a 20 % of involvement in every section. However, this is also a complementary work, because each part of development forms the new component, called ReemComp. The work performed by the author of this project is shown throughout this document. The Table B.1 shows the work distribution of each student.

B.1.1. Goals

The specific objectives of this project can be summarized as follows:

1. Integration of the ROS framework (*Robot Operating System*) beside the NAOTherapist architecture(which uses RoboComp framework), essential to communicate REEM with other components of the architecture.

2. Make the execution of movements in Gazebo simulator with the REEM model through simple commands in Python.

3. From a humanoid skeleton extracted from the Kinect 3D sensor, implement a retargeting of the angles of each joint to obtain correct poses in the REEM robot.

   a) Select the joints of the REEM which allow to do a retargeting (adjusting the value of the angles of the skeleton obtained from Kinect sensor to the REEM angles).

   b) Transform angles to perform the same movement in REEM and Kinect skeleton.

4. Implement a graphical interface that can be represented on the integrated display of REEM. That interface simulates led lights used in NAOTherapist to provide visual information to the person who is interacting with the robot. Also, there are some buttons that simulate physical NAO buttons.

5. Add an audio player and a *Text-to-Speech* system to make the communication among robot and user easier.

6. Integrate the system described in the previous points, applied to an adapted domain of the classic game *Simon*.

B.1.2. Structure of the document

The document has been structured starting from the most general content to the most specific. The first chapter corresponds to the introduction, establishing the context of the project, the main goals that can be accomplished and the structure of the document. Chapter 2 presents the state of the art, including the main theoretical and technical elements of the solutions that have been adopted in this document. Chapter 3 is focused on the analysis and design of the system. This section includes all the specification of system requirements and use cases to understand how the system works. Chapter 4 shows the implementation of the system, including the development of the retargeting system, graphical interface, animations and audio tools.

To prove that the objectives have been accomplished, Chapter 5 contains the evaluation of the system. This section explains the results of several experiments and tests that show if the objectives have been accomplished.

Chapter 6 contains a development planning of the project, in addition to the budget for this work. Finally, Chapter 7 shows the conclusions and future work for this project.

B.2. Evaluation

This section describes the evaluation of the system, where there are some experiments that cover all functions programmed in this work. In addition, these tests are used to demonstrate that the goals outlined in Chapter 1 of this document have been accomplished.

All videos of this evaluation are uploaded on the official channel of NAOTherapist project. Evaluation subsections make reference to this channel videos uploaded to that effect:

https://www.youtube.com/user/NAOTherapist/videos.

B.2.1. Retargeting

In this section, the evaluation of the *retargeting* system is presented. The execution script has been changed in order to launch NAO in Choregraphe simulator and REEM in Gazebo simultaneously. For this test, a set of 7 representative poses have been used. With the retargetingHelper tool, it is possible to set a pose and see the results immediatly in both simulators.

The objective of this test is to evaluate the retargeting method compared to the NAO robot. It is a visual analysis due the morphological differences between two robots, because it is hard to measure the quality of the position in a particular way. In Figure B.1 results are presented.

Seeing the results, some conclusions can be extracted. Due to different morphology on each robot, there are some differences. REEM has considerably longer arms in relation with the body, making that the arms are more open in REEM than NAO, like it is observed in image (2). The head of REEM is smaller than NAO, so the are differences in image (6) for example.

Moreover, not all problems come from to the morphology of the robot. After many transformations in the elbow, the retargeting could still be improved in positions like (4). When shoulderPitch and elbowYaw angles have a value close to 90 degrees there is a slight deviation in the position. Depending on the specific position this deviation is more or less perceived. Response times and movement speed are significantly slower in REEM due to the limitations of the tools provided by PAL Robotics to handle REEM.

After these results it is concluded that the retargeting of REEM works well, although improvements can be implemented. With this experiment, the robot motion and trasformation angles objectives are demonstrated.

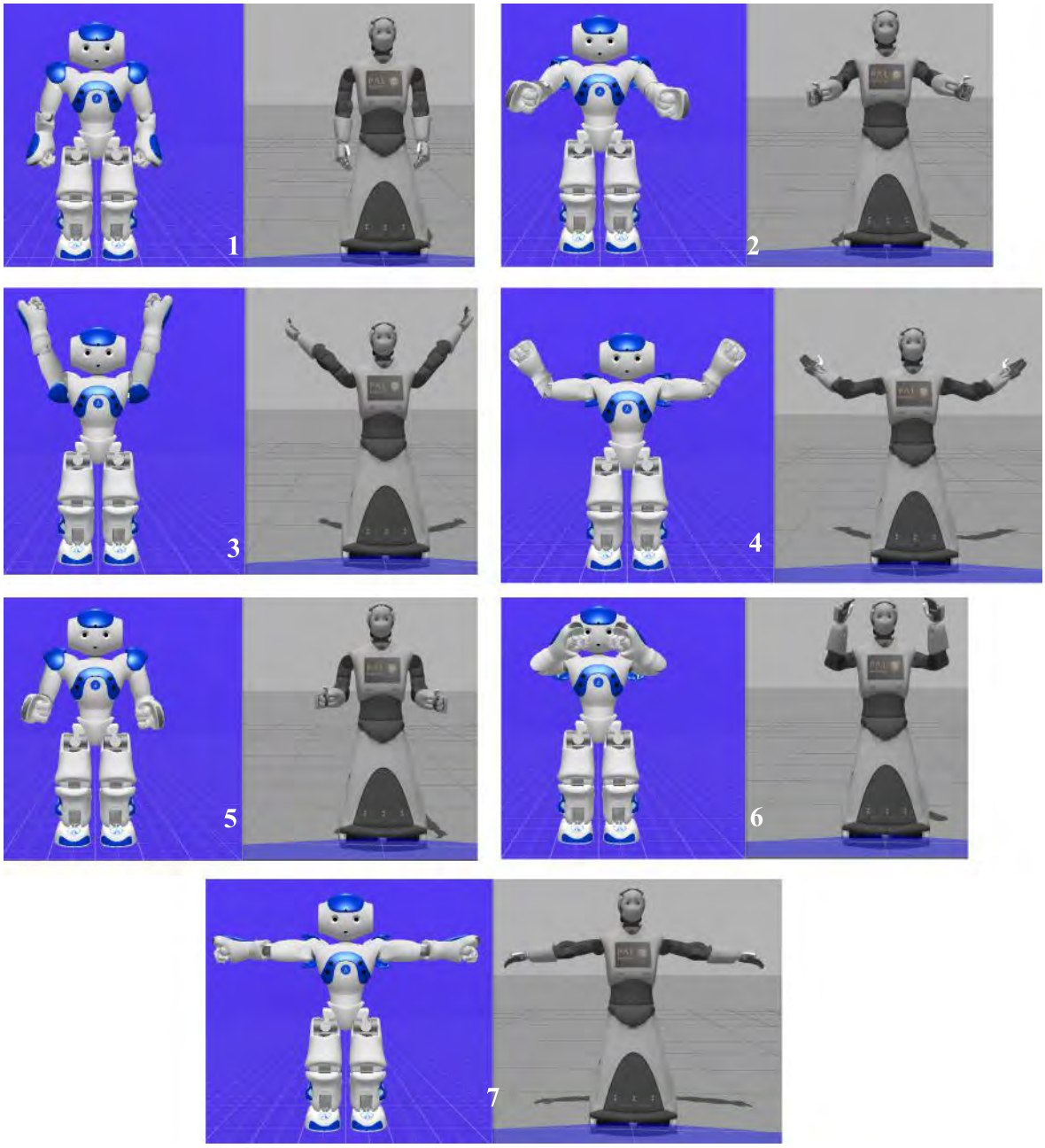

Figura B.1: Retargeting comparison.

B.2.2. Interface

On the one hand, the robot use the leds of the eyes to give feedback to the user, specifying his position is correct or must be corrected. To communicate this,

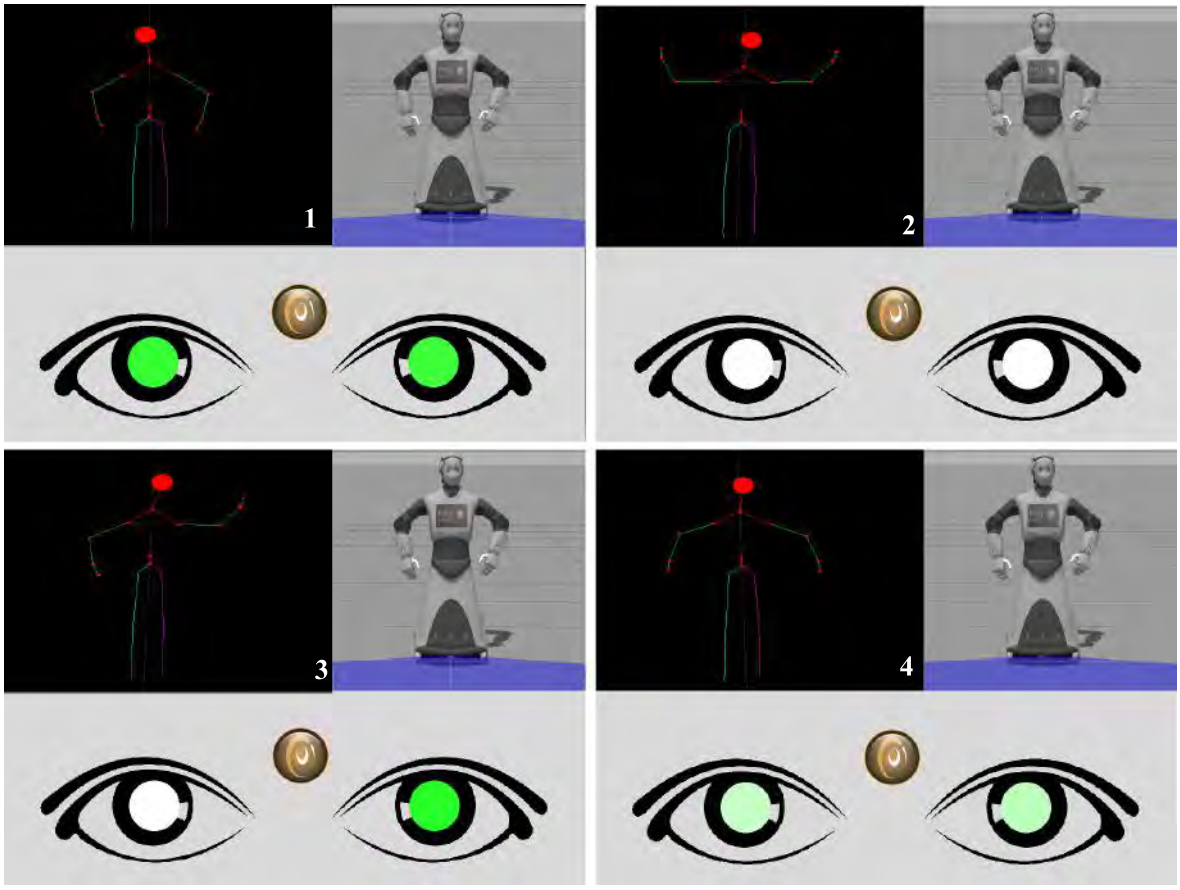

Figura B.2: Evaluation of pose.

the interface has two eyes showing a green color when the position is correct. This green color is brighter when the position is closer to the correct. This visual clue is independent of each arm, so if you hace only one of the two arms misplaced, only one eye reflected the error.

The test was performed using two different poses. The system checks the position of the user located in front of the Kinect sensor and compares it with the pose of the robot, indicating in the interface if it is correct or not. The correct position is shown in the model of the robot, while the posture of the user is reflected in the Kinect skeleton. Results are shown in Figure B.2.

Interestingly, the implementation of this functionality directly on the interface makes the transition between colors much softer than the original architecture with the real leds of the NAO.

One the other hand, to check the functionality of the buttons and demonstrate a proper communication with the rest of the architecture, a test using the *FrontTactil* button is presented. In NAOTherapist, this button pauses the execution of the system when it is pressed. Pressing the button again, makes the execution to continue. The test shows the trace of the Executive component, which shows the break of the session when the button is pressed and resumes the session when it is pressed again.

The following trace is displayed when the button is pressed:

```
Last button pressed set to FrontTactil
->getLastButtonPressed: FrontTactilTouched
```

This action informs the Executive component which button was pressed and executes the corresponding action, which in this case is to pause the execution. The trace for the Executive can be seen below:

```
1 DETECT-PATIENT [‘nao’, ‘pt00’]
2 IDENTIFY-PATIENT [‘nao’, ‘pt00’]
3 GREET-PATIENT [‘nao’, ‘pt00’]
4 START-TRAINING [‘nao’, ‘loc_training’, ‘pt00’, ‘loc_show’, ‘ses10’]
Emergency: 0
5 PAUSE-SESSION [‘nao’, ‘pt0’, ‘ses1’]
6 RESUME-SESSION [‘nao’, ‘pt0’, ‘ses1’]
7 INTRODUCE-EXERCISE [‘e100r’, ‘nao’, ‘pt00’, ‘ses1’]
8 START-EXEERCISE [‘e100r’, ‘nao’, ‘pt00’, ‘stand_up’]
9 EXECUTE-POSE [‘p_id0’, ‘e100r’, ‘p6’, ‘t1’, ‘nao’, ‘pt00’, ‘stand_up’]
```

Each point shown in the trace correspond to actions that the Executive component descomposed into low-level actions (actions that have been programmed into the ReemComp component). The pause button is pressed during the execution of the action *START-TRAINING*, causing the session to pause indefinitely. When the same button is pressed again, the action *Resume Session* is executed and the session continues.

The alternative of using the touch screen to replace the absence of lights in the eyes and physical buttons works fine, so the objectives with the GUI have been accomplished.

B.2.3. Animations and audio player

To perform this experiment, two animations are executed in the context of break points (or rests) in Simon execution.

The result of these tests can be seen in the video **Animaciones - Evaluación REEM** available on the YouTube channel mentioned at the beginning of the section.

This video shows "Do you want to get stronger?" and "Macarena" animations, used in NAOTherapist and originally made by the NAO. These two animations have been selected for this evaluation because they have an audio file included. The experiment demonstrates that both animations and audio work correctly. As a bonus, it is interesting to highlight that audio play is non-blocking, so the song from the Macarena plays while the robot starts to move, for example.

The smoothness of the animation is good as a general rule. However, in faster animations like macarena some transitions between movements are rougher.

B.2.4. Text-to-Speech

For this test a text file is used without a corresponding audio file, so the component must use the *Text-to-Speech* system to play it.

The result of these tests can be seen in the video **Text-to-Speech - Evaluación REEM** available on the YouTube channel mentioned at the beginning of the section.

With this evaluation the objective about audio and animations set in Chapter 1 have been accomplished.

B.2.5. Simon

This test consists of a full execution of the architecture, showing the robot (screen) and the user (located in front of the Kinect).

The result of these tests can be seen in the video **Simon - Evaluación REEM** available on the YouTube channel mentioned at the beginning of the section.

For this test, a modified Simon domain it is used in which we have chosen a set of 3 positions. The full execution shows the game with a real user. This game includes all the functionality of the project and the compatibility of the NAOTherapist architecture with the component developed in addition to the functionality of the domain of the game. Animations and sounds improve the interaction with the user.

In conclusion, indicate that this test really encompasses all others, showing that the new component developed for the architecture works. Therefore, the objectives for this project have been accomplished.

B.3. Conclusions

Once the work is completed some conclusiones can be extracted, noting that the objectives set at the beginning of the document have been accomplished:

- *RoboComp* framework and *ROS* framework are compatible with NAOTherapist architecture, making the whole architecture to work fine.
- It is possible to send commands to the robot through the new component based on data received by the architecture (Kinect skeletons).
- The *retargeting* allows REEM robot have a similar movement to the original skeleton received from Kinect sensor, but the changes made are more difficult than expected.

- The graphical interface communicates with the component created and the rest of the architecture, achieving the main objective by simulating led eyes and physical buttons that the original robot does not have, keeping this functionality through the interface.

- *Text-to-Speech* and audio player works, but *text-to-Speech* has a voice too robotic, subtracting value to the interaction. The *text-to-Speech* have free libraries has a robotic voice, so it is better to use commercial libraries.

- Simon domain implementation works correctly and is quite customizable due to the number of different poses that exists in the architecture. This allows to have multiple versions with different degrees of difficulty.

B.4. Future work

After the finalization of this work, the knowledge of its possibilities and the capabilities of the robot and the architecture, several future research lines can be raised in order to improve or expand this project.

On the one hand, a functional *retargeting* has been made by comparing the position of the robot with the original position of the user. However, there are two factors in NAO robot that can not be handled in REEM: the speed of movement and the non blocking movements. PAL Robotics company reported that there is no method to control these parameters, so a future development line could add this functionality.

On the other hand, having a graphical interface offers many possibilities to add content or customization. Including a menu to select therapies or Simon simulation, also any other domain compatible with the architecture, will make the architecture more complete and accessible for the user.

Another interesting addition is the adaptation of this development to the robot REEM-C[3] (see Figure B.3). This robot is a product of the PAL Robotics company too, but this new model has legs. Using the same arhitecture and sharing most features with the REEM robot used in this project, the adaptation will be quite direct, adding another robot compatible to the architecture.

Figura B.3: REEM-C robot.

[3]http://pal-robotics.com/en/products/reem-c/ Last access 15/02/2016

Bibliografía

[Agravante et al.] Agravante, D. J., Pages, J., and Chaumette, F. Visual Servoing for the REEM Humanoid Robot's Upper Body. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5253–5258, Karlsruhe (Germany). (Citada en pág. 26)

[Alcázar et al.] Alcázar, V., Guzmán, C., Prior, D., Borrajo, D., Castillo, L., and Onaindia, E. Pelea: Planning, learning and execution architecture. In *Proceedings of the 28th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2010)*, Brescia (Italy). (Citada en pág. 15 y 20)

[Alfaro 2012] Alfaro, S. (2012). Sistema de teleoperación mediante una interfaz natural de usuario. *Bachelor thesis, Universidad Carlos III de Madrid.* (Citada en pág. 34, 38 y 40)

[Argall et al. 2009] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483. (Citada en pág. 15)

[Cabibihan et al. 2013] Cabibihan, J. J., Javed, H., Ang, M., and Aljunied, S. M. (2013). Why Robots? A Survey on the Roles and Benefits of Social Robots in the Therapy of Children with Autism. *International Journal of Social Robotics*, 5(4):593–618. (Citada en pág. 3)

[Capek 1921] Capek, K. (1921). *R. U. R. Rossum's Universal Robots.* (Citada en pág. 11)

[Castelli 2011] Castelli, E. (2011). Robotic movement therapy in cerebral palsy. *Developmental Medicine & Child Neurology*, 53(6):481. (Citada en pág. 17)

[Dautenhahn 2007] Dautenhahn, K. (2007). Socially intelligent robots: dimensions of human-robot interaction. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 362(1480):679–704. (Citada en pág. 13)

[Eriksson et al.] Eriksson, J., Matarić, M. J., and Winstein, C. J. Hands-off assistive robotics for post-stroke arm rehabilitation. In *Proceedings of the 9th International Conference on Rehabilitation Robotics (ICORR)*, Chicago (USA). (Citada en pág. 18)

[Fong et al. 2003] Fong, T. W., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and Autonomous Systems*, (42):143–166. (Citada en pág. 3 y 15)

[Fujita 2004] Fujita, M. (2004). On activating human communications with pet-type robot AIBO. In *Proceedings of the IEEE*, volume 92, pages 1804–1813. (Citada en pág. 12)

[Ghallab et al. 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning: theory & practice*. Elsevier. (Citada en pág. 13 y 20)

[González et al. 2015] González, J. C., Pulido, J. C., Fernández, F., and Suárez-Mejías, C. (2015). Planning, Execution and Monitoring of Physical Rehabilitation Therapies with a Robotic Architecture. In *Proceedings of the 26th Medical Informatics Europe conference (MIE)*, volume 210 of *Studies in Health Technology and Informatics*, pages 339–343, Madrid (Spain). (Citada en pág. 1, 18, 65 y 123)

[Graf et al. 2009] Graf, B., Reiser, U., Hägele, M., Mauz, K., and Klein, P. (2009). Robotic home assistant care-o-bot®3 - product vision and innovation platform. In *Proceedings of the Advanced Robotics and its Social Impacts (ARSO), 2009*, pages 139–144, Tokyo (Japan). (Citada en pág. 3)

[Hoffmann 2003] Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341. (Citada en pág. 20)

[Knoblock et al. 1998] Knoblock, C., Barrett, A., Christianson, D., Friedman, M., Kwok, C., Golden, K., Penberthy, S., Smith, D. E., Sun, Y., and Weld, D. (1998). *PDDL - The Planning Domain Definition Language*. (Citada en pág. 92)

[Krägeloh-Mann et al. 2009] Krägeloh-Mann, I. and Cans, C. (2009). Cerebral palsy update. *Brain and Development Journal*, 31(7):537–44. (Citada en pág. 17)

[Limthongthang et al. 2013] Limthongthang, R., Bachoura, A., Songcharoen, P., and Osterman, A. L. (2013). Adult brachial plexus injury: evaluation and management. *The Orthopedic clinics of North America*, 44(4):591–603. (Citada en pág. 17)

[Manso et al. 2010] Manso, L., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., and Calderita, L. (2010). Robocomp: A tool-based robotics framework. In *Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472 of *Lecture Notes in Computer Science*, pages 251–262. Springer Berlin Heidelberg. (Citada en pág. 29)

[Mardiyanto] Mardiyanto, R. 2D Map Creator for Robot Navigation by Utilizing Kinect and Rotary Encoder. In *Proceedings of the 16th Intelligent Technology and Its Applications (ISITIA)*, pages 81–84. (Citada en pág. 32)

[Matarić et al. 2007] Matarić, M. J., Eriksson, J., Feil-Seifer, D. J., and Winstein, C. J. (2007). Socially assistive robotics for post-stroke rehabilitation. *Journal of Neuro-Engineering and Rehabilitation*, 4(5). (Citada en pág. 7)

[Peppoloni et al. 2015] Peppoloni, L., Brizzi, F., Avizzano, C. A., and Ruffaldi, E. (2015). Immersive ROS-integrated framework for robot teleoperation. In *Proceedings of the 10th IEEE Symposium on 3D User Interfaces (3DUI)*, pages 177–178, Arle (France). (Citada en pág. 28)

[Rossignoli 2015] Rossignoli, A. (2015). Desarollo de terapias de rehabilitación motora teleoperadas con el robot NAO. *Universidad Carlos III Madrid*. (Citada en pág. 34, 36, 86 y 88)

[Ruiz et al.] Ruiz, E., Acuña, R., Certad, N., Terrones, A., and Cabrera, M. E. Development of a control platform for the mobile robot Roomba using ROS and a Kinect sensor. In *Proceedings of the IEEE Latin American Robotics Symposium (LARS)*, pages 55–60, Arequipa (Peru). (Citada en pág. 28 y 32)

[Srinivas et al. 2014] Srinivas, C. E. and Mangalore, T. (2014). Kinect Sensor based Real-time Robot Path Planning using Hand Gesture and Clap Sound. In *Proceedings of Circuits, Communication, Control and Computing (I4C) International Conference*, number November, pages 21–22, Bangalore (India). (Citada en pág. 32)

[Suárez Mejías et al. 2013] Suárez Mejías, C., Echevarría, C., Nuñez, P., Manso, L., Bustos, P., Leal, S., and Parra, C. (2013). Ursus: A robotic assistant for training of children with motor impairments. In *Converging Clinical and Engineering Research on Neurorehabilitation*, volume 1 of *Biosystems & Biorobotics*, pages 249–253. Springer Berlin Heidelberg. (Citada en pág. 16)

[Turp et al. 2015] Turp, M., Pulido, J. C., González, J. C., and Fernández, F. (2015). Playing with Robots: An Interactive Simon Game. In *Proceedings of the 16th Conference of the Spanish Association for Artificial Intelligence (CAEPIA), RSIM workshop*, pages 1085–1095, Albacete (Spain). (Citada en pág. 5, 7 y 92)

[Veloso et al.] Veloso, M. and Stone, P. In *Proceedings of the 25th Intelligent Robots and Systems (IROS) IEEE/RSJ International Conference*, Vilamoura (Portugal). (Citada en pág. 13)

[Weinberg et al.] Weinberg, G., Raman, A., and Mallikarjuna, T. In *Proceedings of the 4th Human-Robot Interaction (HRI) ACM/IEEE International Conference*, La Jolla (USA). (Citada en pág. 3)